REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: ulls FILE C
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
AD-A209 242 S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): 24620.17-EG-UIR
6a. NAME OF PERFORMING ORGANIZATION: Massachusetts Institute of Technology, Civil Engineering	6b. OFFICE SYMBOL (If applicable): CCRE/PACT
7a. NAME OF MONITORING ORGANIZATION: U. S. Army Research Office	
6c. ADDRESS (City, State, and ZIP Code): 77 Massachusetts Avenue, Room 1-175, Cambridge, MA 02139	7b. ADDRESS (City, State, and ZIP Code): P. O. Box 12211, Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: U. S. Army Research Office	8b. OFFICE SYMBOL (If applicable):
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code): P. O. Box 12211, Research Triangle Park, NC 27709-2211	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

DTIC ELECTE JUN 19 1989 S E

11. TITLE (Include Security Classification)
An Object Oriented Programming Environment for Communication, Coordination and Control in Computer Integrated Design and Construction: Phase I.

12. PERSONAL AUTHOR(S)
Sriram, D.; Logcher, R.D.; Groleau

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 1/87 TO 12/87	December 20, 1987	37

16. SUPPLEMENTARY NOTATION
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Knowledge-based systems; object oriented programming; Integrated design & construction; blackboard control techniques. (JES)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The development and testing of knowledge based computer tools for the integration of design and construction (CIDCIS) are described. A system architecture is presented which is intended to provide coordination among multiple designers working in separate engineering disciplines, using knowledge to estimate interface conditions between disciplines, recording who used any piece of design data created by others, and how such data was used, and checking for conflicts among disciplines, constructability, and construction cost and schedule impacts of design decisions. The system is based on the object oriented programming and blackboard control techniques. Current status of CIDCIS along with a simulation example is presented.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

An Object Oriented Programming Environment for Communication, Coordination and Control in Computer Intergrated Design and Construction: Phase I

Final Report

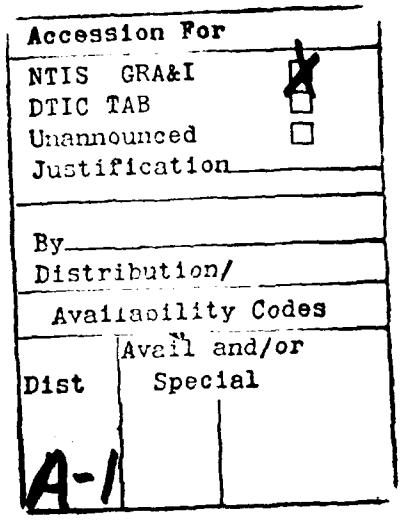
D. Sriram, R. D. Logcher, and N. Groleau

20 December 1987

Contract/Grant Number DAAL038 7K0005

U. S. Army Research Office

Department of Civil Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139



Approved for public release; Distribution unlimited

Table of Contents

List of Figures

List of Tables

An Object Oriented Programming Environment for Communication, Coordination and Control in Computer Intergrated Design and Construction: Phase I

Abstract

The development and testing of knowledge based computer tools for the integration of design and construction (CIDCIS) are described. A system architecture is presented which is intended to provide coordination among multiple designers working in separate engineering disciplines, using knowledge to estimate interface conditions between disciplines, recording who used any piece of design data created by others, and how such data was used, and checking for conflicts among disciplines, constructability, and construction cost and schedule impacts of design decisions. The system is based on the object oriented programming and blackboard control techniques. Current status of CIDCIS along with a simulation example is presented.

1 Introduction

On July 17, 1981, two skywalks in the lobby of the Hyatt Regency Hotel in Kansas City collapsed. Cited as the "most devastating structural collapse ever to take place in the United States", 113 people died and 186 were injured [11]. This was not only a failure of a physical structural system, but also a failure of the process by which most projects in the U. S. are designed and built. The primary objective of our research is to provide computer based tools which would help during design and construction to avoid errors of the type made in Kansas City.

The Hyatt failure was attributed to a combination of three events. First, in progressing from the preliminary to detailed design, where joint and connection detailing occurs, the design of the hanger to spandrel beam connection was inadequate. Second, in developing shop drawings, the connection detail was changed by the steel fabricator, thereby "compounding an already critical condition." Third, this second error was not caught during approval checking of the shop drawings by the structural engineers. These were all errors of communication and coordination in the design process, errors caused by the structure of the process, lack of tools used in this process, and focus on documenting the product of design while neglecting "process" and "intent" documentation.

Construction creates, in general, one-of-a-kind products which are unique configurations of widely used components. The planning process is a typical configuration type process. Because of the large number of components and the interactions of multiple technologies, the components included in the product are decided in an iterative design process. In each iteration, interfaces and interface conditions among these components are designed with slack to account for potential variations created when the components and interface values become better known. Iteration proceeds towards increasing detail; design personnel may change, and their numbers expand with increasing level of detail.

Construction in the U. S. is fragmented. On a single project, interacting design technologies come from separate firms, and there is little coordination between designers and contractor(s). Because designers find coordination among themselves difficult, they leave this task to construction managers or the contractor. Thus, working drawings lack detail. Shop or fabrication drawings are required to document details, but potential conflicts among trades are often unrecognized until construction begins. Several undesirable effects are caused by this lack of coordination.

1. The construction process is slowed, work stops when a conflict is found.
2. Prefabrication opportunities are limited, because details must remain flexible.
3. Opportunities for automation are limited, because capital intensive high speed equipment is incompatible with work interruptions from field recognized conflicts.
4. Rework is rampant, because field recognized conflicts often require design and field changes.

5. Conservatism prevades design, because designers provide excessive slack in component interfaces to avoid conflict.
6. The industry is unprepared for the advent of automated construction, as the need for experience in design limits choice to available materials placed by hand.

All of these problems decrease productivity. In addition, failures, such as the Hyatt collapse, occur more often then they should. Overcoming these problems requires significant changes to the design process, together with superior *computer integrated construction* (CIC) tools. Those tools must be tailored to the needs of designers who are [2]:

"constantly engaged in searching out various consequences of design decisions [especially those made by others]"

This report details the development of a prototype system to test new concepts for computer tools to integrate design and construction. The major objectives of our research are to:

1. Facilitate effective coordination and communication in design and construction.
2. Capture the process by which individual designers make decisions, that is, what information was used, how it was used and what did it create.
3. Forecast the impact of design decisions on construction.
4. Provide designers interactively with detailed construction planning.
5. Develop intelligent interfaces for automation.

Our framework for a computer integrated design and construction system (CIDCIS) could significantly improve productivity by,

- reducing error in design;
- providing more detailed design;
- providing better construction planning;
- allowing easier recognition of design and construction problems;
- using constructability criteria throughout design; and
- advancing automation.

Lessons from the Hyatt failure show that such tools are required. Had the connection designer had access to the concepts of load transmission underlying the preliminary design, local buckling might have been recognized and the joint details changed. Had the fabricator preparing the shop drawings had access to that information, he would have seen that his change violated the purpose of the connection scheme. Had the shop drawing checker seen all these changes together with their intent and known, he would have recognized the faults in the design.

The engineering design process and problems associated with this process are described in Section 2 . Using this background, an overview of the proposed product of this work is given. Background material on computer-based technologies used in this work is presented in Section 3. A system architecture which utilizes concepts from knowledge-based systems and database management systems is described in Section 4. This is followed by a description of the current status of the project and a list of publications.

2 Scope of Work

The problems that engineers normally solve fall along the *derivation-formation* spectrum [1]. In derivation-type problems, solutions consist of identifying an outcome or hypothesis from a finite set of outcomes known to the problem solver. By contrast, in formation-type problems, the problem solver has only the knowledge of how to form the solution. A variety of problem solving techniques are invoked to arrive at a solution.

Design and construction planning problems fall at the formation end of this spectrum. Design and Planning are accomplished by a team of engineers, each knowledgeable in a particular aspect of the problem, but with little knowledge of the decision processes of others. Each could be considered as one of many sources of knowledge, and hence, design and construction could be viewed as *a process of constructing an artifact which satisfies constraints from many sources by using knowledge which also comes from many sources*. The extent of interactions can be seen by looking at the diverse set tasks, listed in Table 1, that must be performed by a diverse set of professionals during the design, for example, for a high rise building [14].

Planning	Architectural design
Spatial layout	Site planning
Preliminary structural design	Analysis modeling
Component design	Geometric modeling
Substructure design	Cost estimating
Electrical distribution design	Elect. distribution analysis
Mechanical design	Mechanical analysis
HVAC design	HVAC analysis
Vertical transportation design	Regulatory compliance
Various design critics	Fire safety analysis

Table 1: Tasks involved in the Design of a Tall Building

As CAD/CAE becomes more widespread, each of these consultants will be performing increased amount of their work with computer tools, tools which will embody and use their knowledge in their speciality area.

From this view, the stages of structural design and construction might be described as [15]:

1. **Conceptual design** involves the selection or synthesis of a potential (preliminary) design satisfying a few key constraints.
2. **Analysis** is the process of modeling the selected system and determining its response to external effects.
3. **Detailed design** is the selection and proportioning of components such that all applicable constraints are satisfied.
4. **Design Review** involves evaluation of the detailed design, produced above.
5. **Construction** involves the preparation of shop drawings, development of detailed construction schedules, actual construction, and construction monitoring.

During each stage in this process, representatives from the various interacting disciplines meet and discuss potential interactions between the components they are designing. They use estimates of space needs, structural, heat, and electrical loads, and other factors to set requirements for their systems based on the needs of others. Experience is used to estimate these interfaces. Explanations on how these estimates were determined is seldom sought, except where they cause conflicts between objectives. When individual designers select components and systems during any stage of design, they use and try to develop solutions which satisfy the interface estimates.

The problem with this process is that individual designers often lack sufficient experience in both estimating their interfaces (assessing their impact on others) and in asking for information needed from others. They assume, instead, situations similar to other designs. Similarly, they seldom think about and may even lack knowledge of constructability or management and control of the construction process. This may lead to incompatible component selection and poor choice of design parameters. For example, the use of wide rooms in low cost housing is incompatible with inexpensive construction techniques. The designer is assumed in this process to have sufficient knowledge of construction techniques, materials, and equipment to make proper decisions. This is seldom the case.

Also, since the present design process does not document reasons behind decisions, others cannot easily question decisions or improve designs.

In this report we will provide a conceptual framework for a computer-based system that addresses the above problems. Figure 1 provides a user view of a CIDCIS (Computer Integrated Design and Construction System), where users within their discipline interact with individual CAD tools and KBS for component design and solution generation. These systems automatically communicate with a global system which provides data and support facilities.

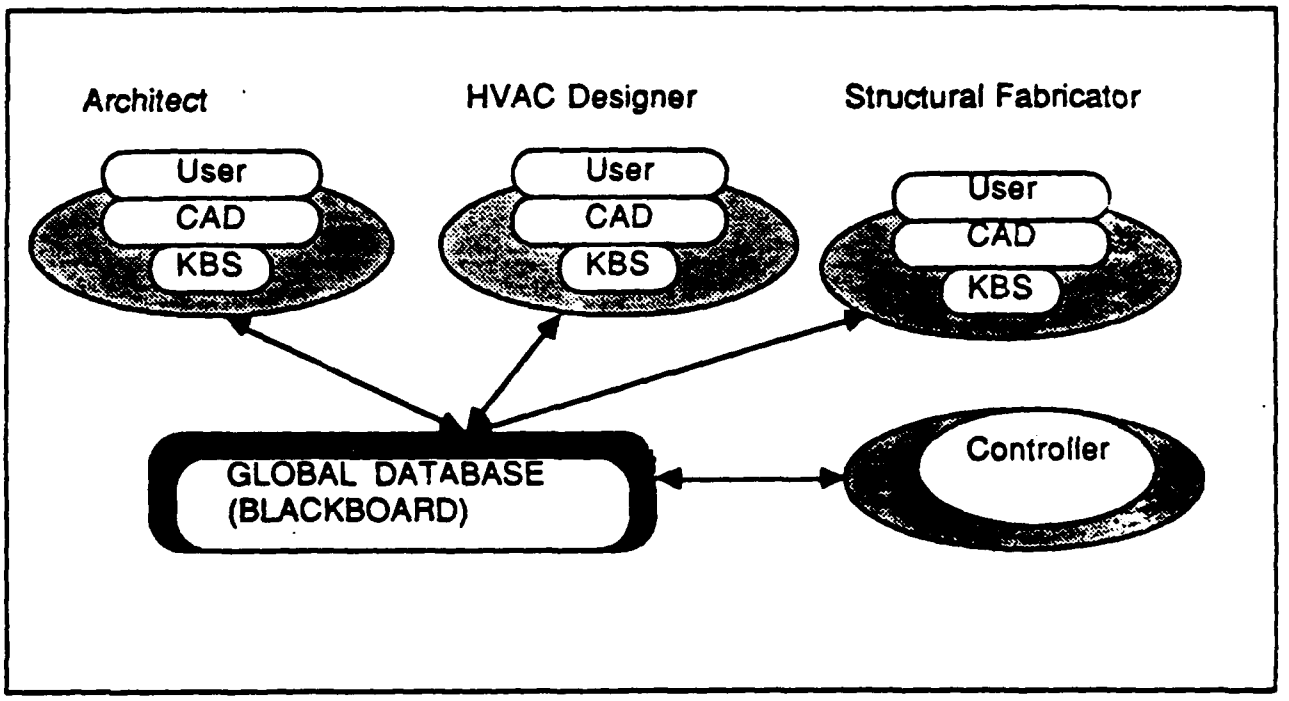


Figure 1: User View of CIDCIS

In the first version, we plan to use two interacting computers, one operating with a commercial CAD system, the other a high speed workstation with good knowledge representation tools. Knowledge representations for support facilities are required to:

1. Estimate and negotiate interface parameters between stages of design, doing so in an interactive manner, when a designer asks for information (i.e., if a designer asks for information that has not yet been developed, knowledge will be used to estimate values);
2. Keep track of who used design information, when, and whether it was estimated or actual values (so that when values change, the design can remain coordinated);
3. Use coded individual knowledge sources to assist in or automate component design, retaining component information about sources of data used in the design, the algorithms or knowledge used, and inputs on design rationale from the user;
4. Operate numerous background processes to check design choices for interferences, violations of interface assumptions, constructability, and cost and schedule impacts;
5. Allow user input and design alterations from either the CAD system or the knowledge representation workstation; and
6. Inform designers of the impacts of initial designs and changes by others on their design choices.

Development of this system involves solutions and integration of the following basic requirements for managing the complexity of engineering design knowledge and data [10].

1. Hierarchical knowledge/data structure.
2. Object oriented representations.
3. A mechanism that keeps tracks of design and planning justifications.
4. Multiple, correlated design requirements.
5. Cross representation consistency checking.
6. Version control.
7. Keeping track data use.
8. Programming language interfaces.
9. Interaction with graphics and drafting.
10. Conventional databases and analysis tools.
11. Integration with construction (manufacturing) project management data.

A framework that addresses most of the above issues is described in Section 4.

3 Background on Computer-Based Technologies

Several computer-based technologies required for this work work are briefly described in Sections 3.1 and 3.2.

3.1 Relevant Computer-Based Technologies

Developments in computer science and engineering methodologies have provided engineers with a variety of software development tools. The computer-based software development tools that are relevant to this project are:

1. Object Oriented Programming (OOP) Methodologies;
2. Knowledge based systems (KBS);
3. Database management systems (DBMS); and
4. Traditional algorithms.

Object Oriented Programming. Object Oriented Programming (OOP) is a style of programming that involves the use of objects and messages. Objects are defined by Stefik and Bobrow as [19]:

> Objects are entities that combine the properties of procedures and data since they perform computations and save local state.

Objects contain slots and slots may consist of a number of facets. A slot may simply be an attribute or it may be a relation. The facets contain meta-information about the slot.

All actions in object oriented programming are performed through messages. Messages tell the object *what to do* and not *how to do it*. Methods are attached to the object to execute the actions associated with the messages. The message passing ability in OOP supports the concept of *data abstraction*.

Objects can be grouped into "classes," where each class of objects knows how to perform several actions. Individual instances of objects can be created from a particular class. The Object Oriented programmer builds a system by specifying new classes of objects and their associated methods. Most OOP systems support the concept of "inheritance," where a class of objects may be specified as a "subclass" of another "superclass" of objects. Subclasses and instances inherit methods from their superclass, and are usually more specific entities than their (usually) more general superclass. An object may inherit methods and data from multiple classes through a network of structural relationships. In short, every object has the ability to: *store information, process information, create*

new information, and communicate with other objects. Thus OOP facilitates encoding design and construction knowledge in a disaggregated and modular form.

As an example consider the following object:

```
BEAM-1
  instance: "Beam"
  M :
  Methods: Display-moment, Calculate-moment
```

The message *(send beam-1 calculate-moment)*, where beam-1 is the object to which the message is addressed, would compute the moment in accordance with the Calculate-moment method. For further details about object oriented programming see [19].

Knowledge-based systems. KBS are computer programs which incorporate knowledge and reason through the application of their knowledge to data about a specific problem. If these systems incorporate human expertise then they are called knowledge-based expert systems (KBES). For the purpose of this proposal, the term KBS will also be used to connote KBES. A typical KBES consists of three components: *Knowledge-base, Context,* and *Inference Mechanism or Control Mechanism.* Several problem solving architectures used in the Inference Mechanism are described in [18]. In this work, a variation of the Blackboard architecture, which facilitates the integration of diverse sources of knowledge through a global database - the Blackboard, will be used [6]. A brief description of the Blackboard architecture is provided in the next section. In addition, the work on truth maintenance systems will also be utilized [3, 4].

Database management systems. Engineers have always dealt with large amounts of data in diverse applications. Hence, storing and manipulating data forms an integral part of the engineering process. Database management systems (DBMS) provide means to store large amounts of data for use by a variety of applications. Data access is controlled through a dictionary so that individual programs need not be changed when the database structure changes.

A number of systems that utilize some of the computer-based technologies and relevant to the proposed work are described in the following sections.

3.2 Blackboard Architecture & Relevant Systems

The Blackboard architecture provides a framework for: 1) integrating knowledge from several sources, and 2) representing multiple levels of problem decomposition. It uses two basic strategies [12]: 1) divide and conquer, and 2) opportunistic problem solving. The divide and conquer strategy is realized by decomposing the context, which is called a *Blackboard*, into several levels depicting the problem solution decomposition, while opportunistic problem solving is achieved by focusing on the parts of the problem that seem promising. The Blackboard architecture has been successfully used in solving a wide range of tasks, such as speech recognition [5], signal processing [13], and planning [6]. In this section, an overview of the Blackboard architecture is presented[1].

A Blackboard system consists of a number of *Knowledge Sources* that communicate through a *Blackboard* and

[1]The article by Penny Nii in the Summer 1986 issue of the AI Magazine provides a good overview and survey of several Blackboard based KBES.

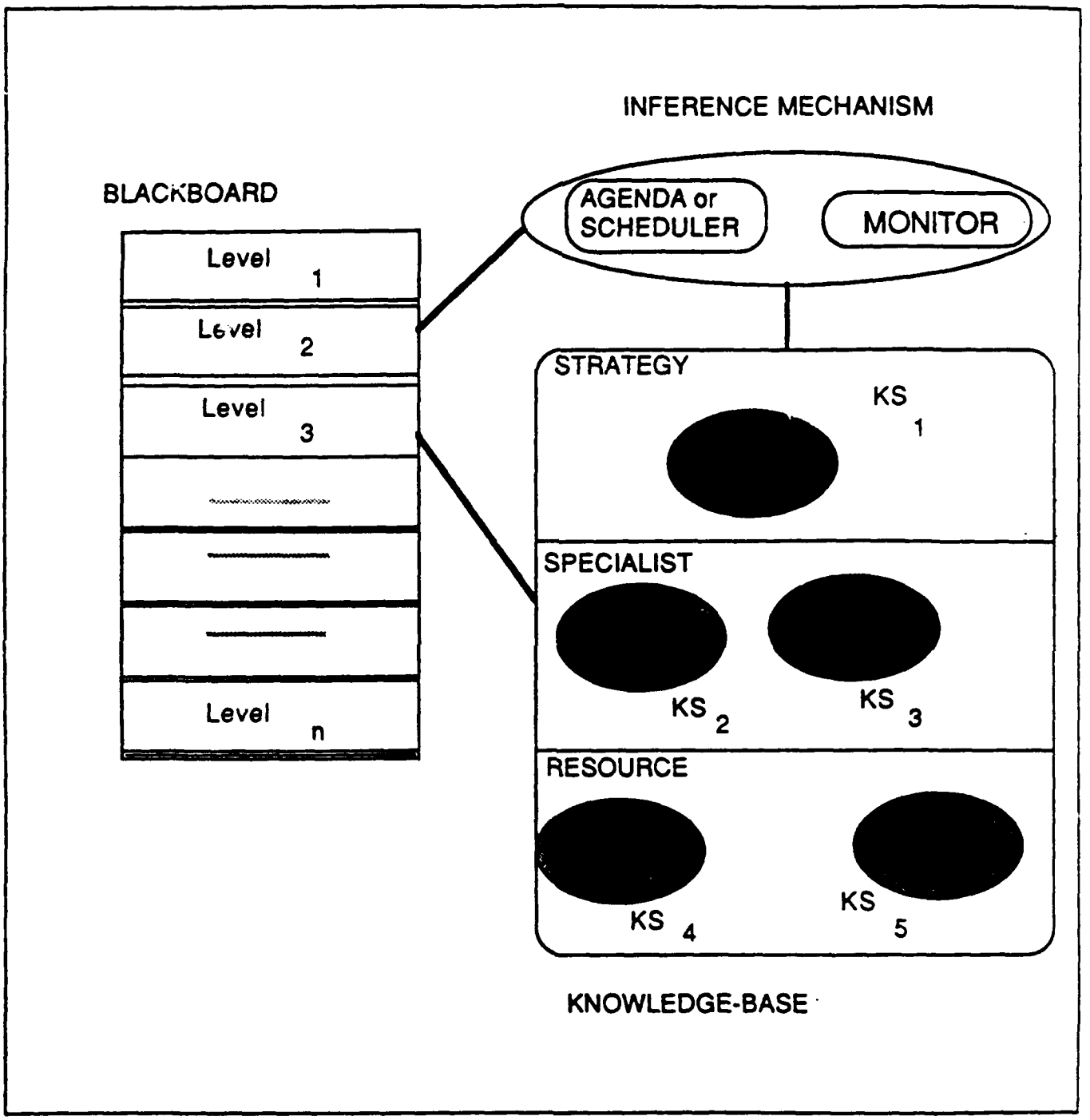


Figure 2: The Blackboard Architecture
(Adapted from [7])

are controlled by an *Inference Mechanism*, as shown in Figure 2. These components are described below.

Knowledge-Base. The Knowledge-base consists of a number of knowledge sources (KSs). These KSs are independent *chunks* of knowledge and do not directly communicate with each other. Instead, they participate in the problem solving process by creating entries in a global database - the Blackboard. Each KS consists of a condition-action pair. When the conditions of a particular KS are satisfied the KS is placed on an agenda in the Inference Mechanism. The actions of the KS are executed when the KS becomes executable, i.e., it has a high priority rating. KSs can also be organized into larger chunks called knowledge modules (KMs). The knowledge-base can be further organized into various levels, as shown in Figure 2; this organization was first implemented in HASP/SIAP [13]. These levels depict a plan for organizing problem solving activities.

Blackboard. The Blackboard or Context consists of the information or entries generated by the KSs during the problem solving process. It is organized into a number of levels. These levels depict an *a priori* plan for the solution of a problem that can be naturally decomposed into a set of levels. Each level contains objects and attributes that are important to the representation of the problem. The hierarchy of levels in the Blackboard is known as the *abstraction hierarchy*.

In addition to the vertical abstraction hierarchy, the Blackboard can also have a horizontal dimension. For example, in HEARSAY-II and OPM, the horizontal dimension represents overlapping intervals in the solution. Normally, KSs are specific to certain levels in the Blackboard, i.e., the activation of a certain KS depends on the entries generated at certain levels in the Blackboard, while the actions of the KS modify entries at some other level.

The main units in the Blackboard are hypotheses. The hypotheses are either primary guesses about particular aspects of the problem or partial solutions. Hypotheses at various levels are related through structural relationships.

Inference Mechanism. The control strategy incorporated in various Blackboard-based systems differ in many aspects. In the earlier versions of Blackboard systems, the Inference Mechanism consisted of two main components: the *Agenda or Scheduler*, and the *Monitor*. The Agenda keeps track of all the events in the Blackboard and calculates the priority of execution for KSs that were generated as a result of the activation of other KSs. The Monitor takes the element with the highest priority and executes it. Several problem solving strategies can be implemented using the Agenda and the Monitor. The Inference Mechanism described above was further elaborated in several Blackboard-based KBES.

A summary of current Blackboard-based systems is provided in Table 2. Further information about these systems can be found in [9].

4 A Framework for Computer Integrated Design and Construction

In Section 1 (page 2) several objectives for a Computer Integrated Design and Construction System (CIDCIS) have been enumerated. To achieve these goal, a system architecture based on current trends in programming methodologies, object oriented databases, and knowledge based systems is described. An overview of a CIDCIS is provided in Section 4.1. This is followed by a discussion of various components comprising the system.

4.1 Overview of CIDCIS

CIDCIS can be envisioned as a network of computers and users, where the communication and coordination is achieved, through a global database, by a control mechanism. CIDCIS consists of several Knowledge Modules, a Blackboard, and a Control Mechanism. These terms are clarified below.

1. **Control Mechanism.** The communication, coordination, data transfer, and all other functions define

System	Domain	Characterisitcs	Limitations For CAE (Lack of ..)	Relevancy to Proposed Work	Developer (Date)
HEARSAY-II	Speech understanding	Knowledge centered, sophisticated scheduler, some temporal reasoning	Object semantics, proper interfaces between KSs and BB, support for parallel computation consistency maint-enance	BB levels	Erman, Reddy, Lesser, Hayes-Roth (CMU, 1975)
HASP/SIAP	Signal processing	Node centered, knowledge levels, some temporal reasoning		BB levels, KS organization	Nii, Feigenbaum, (Stanford, 1978)
BB1 (OPM)	OPM-planning, BB1 is a domain independent system	Control and domain BBs, control and domain KSs, sophisticated problem solver	Proper interfaces betwn. KSs and BB Parallel computation	BB levels	Hayes-Roth(Barbara) (Stanford, 1983)
POLIGON	Domain independent system that supports parallel computation	Targeted for a MIMD, distri-buted machine, message passing machine	Proper interfaces betwn. KSs and BB application to complex engineering tasks, consistency maintenance, support for CAD databases, etc..	Parallel computation (some ideas)	Rice (Stanford, 1986)
ERASMUS	Domain independent system similar to BB1	Similar to BB1. Can also con-figure distributed BBs.		Distributed computation	Jagannathan, Dodhiwala, Baum (Boeing, 1986)
CASSANDRA	Facilitates distributed computation	Level managers, matcher, scheduler, interperter for each level in BB.		Same as above	Craig (Warwick, UK, 1987)
CODGER	Mobile robot navigation	Parallel architecture, distributed KSs, real time procesing	Same as above	Same as above	Shafer, Stentz (CMU, 1987)
DSPK	Model organizatons of distributed, collaborating programs	Network of computers, shared memory	Same as above	Same as above	Talukdar, Cardozo (CMU, 1987)

(Note: Development dates are approximate)

Table 2: Summary of Various Blackboard Systems

the Control Mechanism. Sometimes this control mechanism is also known as Inference Mechanism.

2. **Blackboard.** The medium through which all communication takes place. The Blackboard in the proposed system is divided into three partitions: Coordination, Solution, and Constraint Definition. The Solution Blackboard partition contains the design and construction information generated by various Knowledge Modules, most of which is referred to as the Object Hierarchy containing information about the design product and process, while the Constraint Definition Blackboard partition contains the interaction (or interface) constraints between objects depicting various components of the design and construction process. The Coordination Blackboard partition will contain the information needed for the coordination of various Knowledge Modules.

3. **Knowledge Module.** A Knowledge Module can be viewed either as: a knowledge based expert system, developed for solving individual design and construction related tasks, or a CAD tool, such as a database structure, i.e., a specific database, an analysis program, etc., or an user of a computer, or a combination of the above. A KBES could be viewed as an aggregation of Knowledge Sources (KSs). Each KS is an independent chunk of knowledge, represented either as rules or objects. In the proposed system, the Knowledge Modules are grouped into three categories: Strategy, Specialist, and Resource. The Strategy KMs help the Control Mechanism in the coordination and communication process. The Specialist KMs perform individual specialized tasks of the design and construction process, while the Resource KMs are mostly algorithmic CAD tools.

A conceptual view of a CIDCIS is shown in Figure 3. In it, any of the KMs can make changes or request information from the Blackboard; requests for information are logged with the objects representing the information; and changes to the Blackboard may initiate either of the two actions: finding the implications and notifying various KMs, and entering into a negotiation process, if two or more KMs suggest conflicting changes.

Details of individual components are provided in the following sections.

4.2 Control Mechanism

The Control Mechanism performs two tasks: 1) evaluate and propagate implications of actions taken by a particular KM; and 2) assist in the negotiation process.

Task 1 is accomplished through:

1. methods associated with objects in the Object-Hierarchy of the Solution Blackboard partition (SBB); and

2. a truth maintenance system which keeps the global database in a consistent state.

If two KMs try to access the same object, then the priorities are achieved by the Strategy KM and the scheduling information is stored in the Coordination Blackboard partition (CORDBB). A possible trace of events is shown in Figure 4, and outlined below.

1. A preliminary design of a building (in the form of objects) which includes loading details and designer's intentions in making certain decisions is posted on to the Solution Blackboard partition by the Conceptual Designer.

2. Let the connection details of a particular joint be represented by the Connection object. The Connection Designer will send a message with details of connections and any assumptions made during the design.

3. The truth maintenance system checks to see whether earlier assumptions made by the Conceptual Designer are violated or not.

4. Associated with the Connection object are methods, which indicate the possible KMs that can modify the object. Assume that Fabricator KM is one of them. A message is sent to Fabricator KM to find out whether the connection can be fabricated in the field.

5. Notify the Connection Designer if any problems are anticipated.

6. Sometimes two or more KMs may want to modify or access a particular object in the Solution

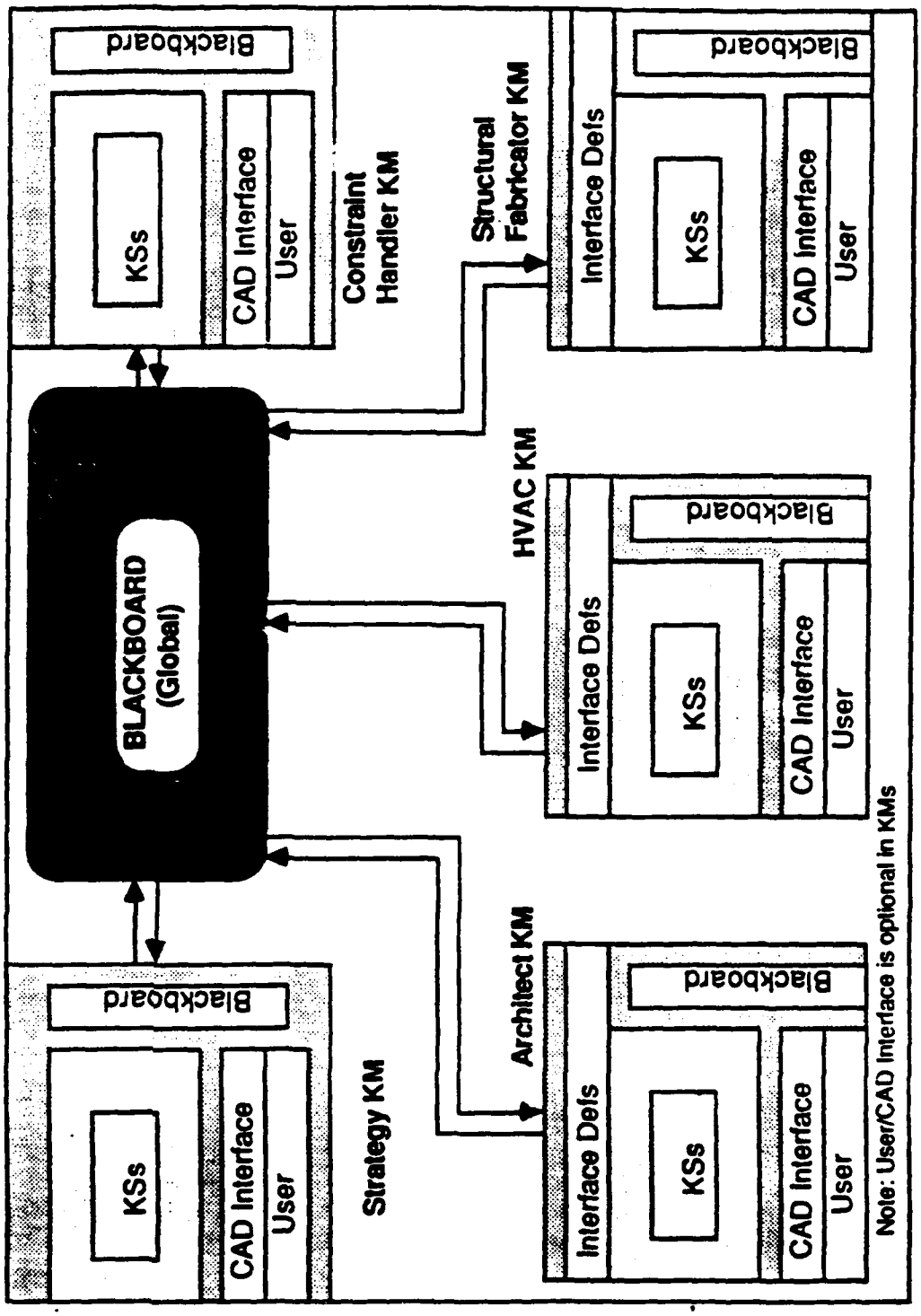


Figure 3: A Conceptual View of CIDCIS

Blackboard partition. This information is stored in Coordination Blackboard partition and is used by the Control Mechanism.

A possible scenario for task 2 for a domain which involves the design and construction of interior finishes is given below (See Figure 5).

1. Let the Architectural KM post the location and other details of beams in the beam object, whose primary owner is Architectural KM.
2. The HVAC (Heating, Ventilating and Air Conditioning) KM would post a message with the assumption that there is a hole in the beam.
3. Solution Blackboard partition sends a message to Constraint Definition Blackboard partition, which checks to see if the objects being modified have any interaction (interface) constraints. If so then appropriate constraints are stored in the Coordination BB partition.
4. Solution Blackboard partition, then, sends a message to Strategic KM to check the constraints.
5. Strategic KM sends a message to the Constraint Handling KM (CHKM). CHKM checks to see if the interaction (interface) constraints are satisfied. If so, a message is sent to Solution Blackboard partition and appropriate actions are taken (step 6)
6. If the interaction constraints are not satisfied then the Strategic KM performs a constraint negotiation. Constraint negotiation may involve relaxing constraints by a particular KM. If constraint negotiation fails then system goes into a deadlock and alerts the KMs. Constraint negotiation can be performed at several levels. In the current system it will be assumed that refinement of levels in the Solution Blackboard partition occurs only after appropriate interaction (interface) constraints are satisfied.
7. If above process succeeds then Strategic KM sends a message to Solution Blackboard partition, at which stage the details required for the next level in the Solution Blackboard partition are set up and appropriate KMs are activated.

4.3 Blackboard: Global Database

The purpose of the Blackboard is to: 1) provide a means for storing information that is common to more than one KM; 2) facilitate communication and coordination; and 3) ensure that designs and plans generated during design and construction are consistent.

The Blackboard in the proposed CIDCIS will be partitioned into: Coordination (CORDBB), Solution (SBB) and Constraint Definition (CDBB) (Figure 6).

4.3.1 Coordination Blackboard Partition

The Coordination Blackboard partition (CORDBB) contains the bookkeeping information needed for the coordination of KMs.

4.3.2 Solution Blackboard Partition

The Solution BB partition (SBB) is divided into levels (object hierarchy). Each level contains objects that represent certain aspects of the design and construction process. For example, the 3D space level contains objects that represent spaces allocated to structural systems, piping systems, mechanical systems, etc. This level can be reduced to detailed levels, such as system and component levels.

The objects in SBB are connected through relational links, where the relational links provide means for objects to inherit information; these relationships provide a framework to view the object from different perspectives. In this work, the following relationships are needed in the SBB: *generalization (IS-A)* for grouping objects into class, *classification (INSTANCE)* for defining individual elements of a class, *aggregation (PART-OF, COMPONENT)* for combining components, *alternation (IS-ALT)* for selecting between alternative concepts, and *versionization*

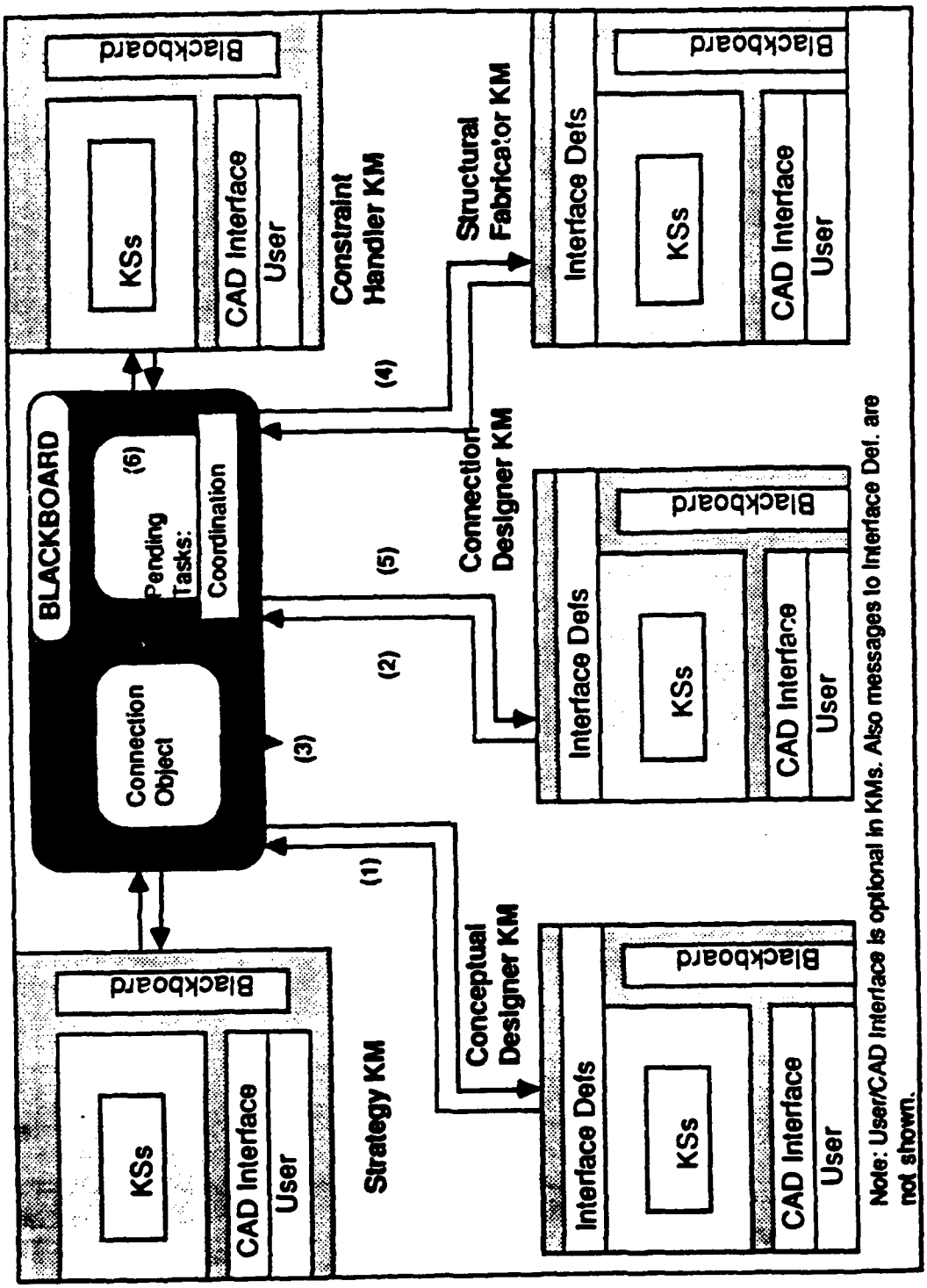


Figure 4: Evaluation and Propagation of Implications

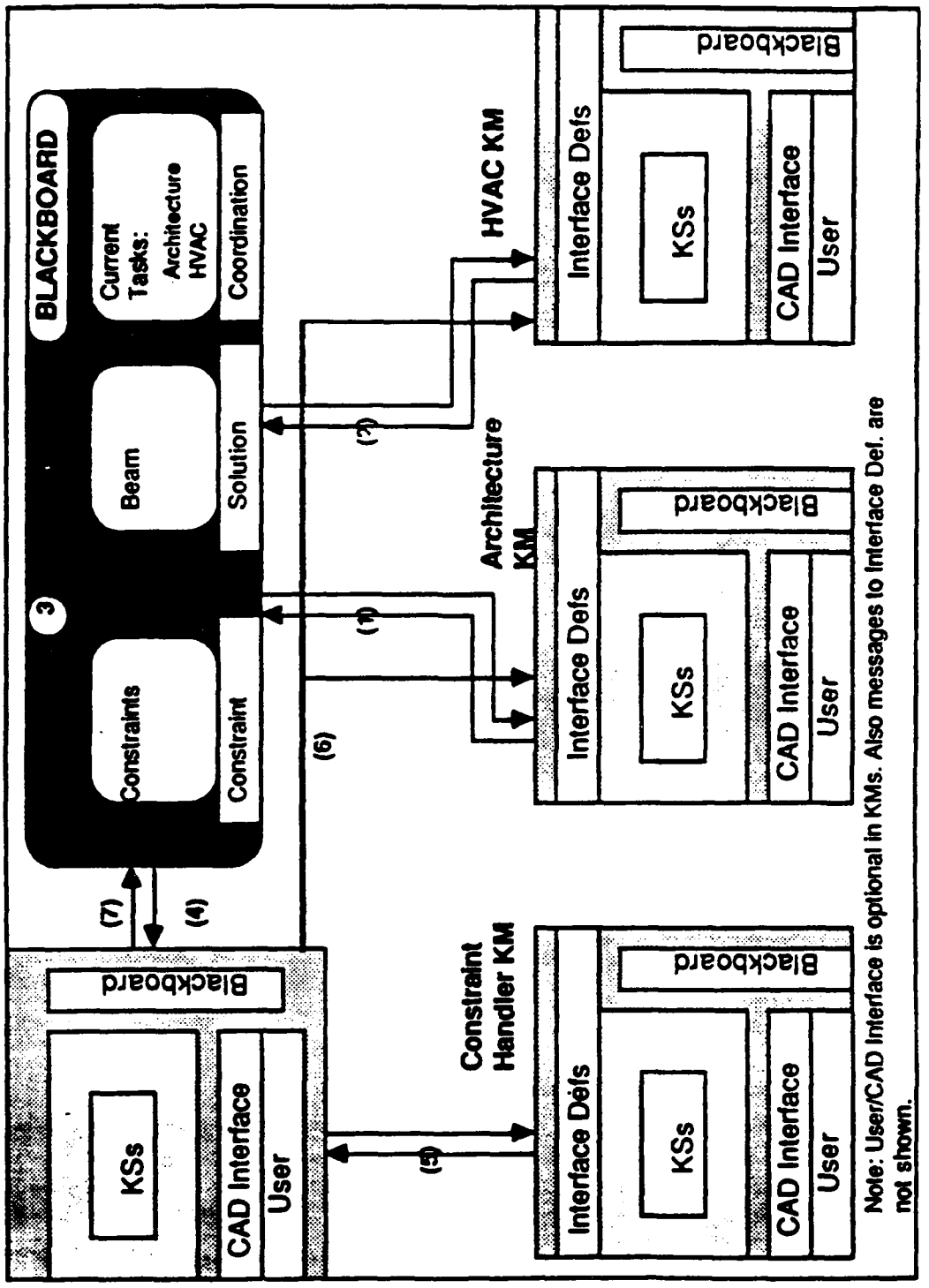


Figure 5: The Constraint Negotiation Process

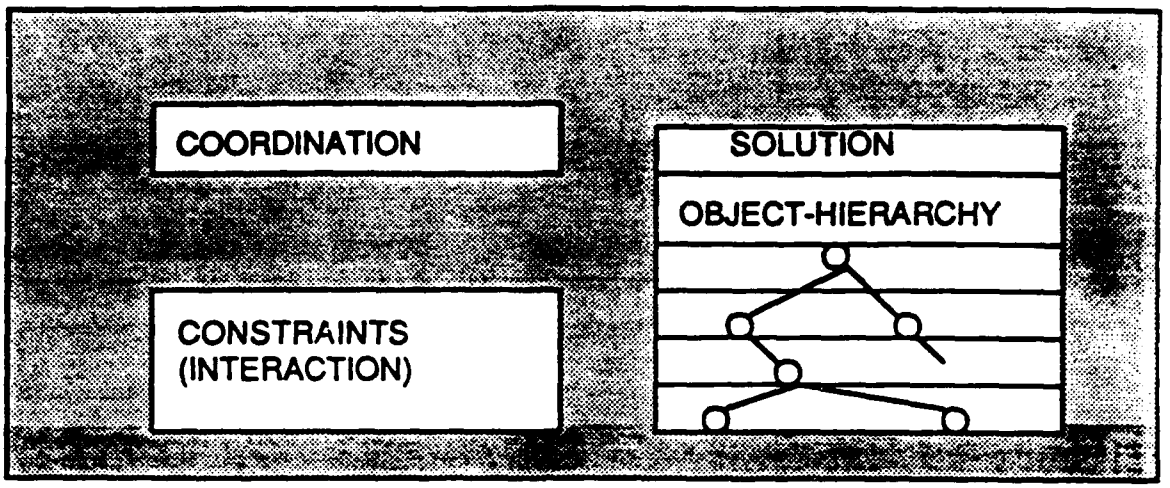


Figure 6: The Blackboard

(VERSION-OF) for representing various versions of an object. The semantics of these relationships are provided in [17].

The objects also contain justifications, assumptions, time of creation, creator, constraints, ownership KM, other concerned KMs, etc. The justification information will provide a designer's rationale and intent for the creation of the object. Assumptions made during design and construction are also stored with the object. For example, the architect, while placing the structural elements, may assume certain spatial characteristics for the HVAC systems. He may record this assumption and the rationale for such an assumption in the objects denoting the appropriate structural elements and the HVAC system. In CIDCIS, *status* facets are associated with data attributes (slots). The *status* facet, for example, can take the following values: *unknown*, *assumed* and *calculated*. Additional slots may be needed for the source of data and its change, uses of data, assumptions made, etc..

Associated with these objects are methods which provide a means for: 1) performing some procedural calculations; 2) propagating implications of performing some actions, for example if the status (assumed or actual) or the value for a particular object changes then these changes can be broadcast to all concerned KMs; 3) helping to perform the coordination process. For example methods can be used as demons to perform the following construction related tasks:

1. **Estimating**, which involves continuous cost forecasting capabilities, from early estimates to detailed costs considering the equipment that will be available. This estimating will start with material and quantity modeling based on building standards for tenant work, and would first be updated with characteristics of the tenant. As layout work proceeds, material and quantity estimates would be updated.
2. **Scheduling**, which is similar in structure to Estimating, and uses much of the quantity data developed from the estimate forecast, passed to it with messages.
3. **Constructability**, where constant critics look for incompatible materials, space use, construction space

needs, equipment requirements, etc.

Knowledge for all of these inputs will come from working with expert on all phases of the project, owner, designer and constructor. Further details of the use of methods in the communication process are provided in Section 4.2.

A typical object that resides in the SBB could be structured as follows:

```
SBB-Object
        NAME:
        VALUE:
                status:
        CREATED-BY:
        JUSTIFICATION:
        PART-OF:
        IS-ALT:
        VERSION-OF:
        VERSION-NO:
        OWNED-BY:
        CONCERNED-KMS:
        CONSTRAINTS:
                range: (IS-A CONSTRAINT-OBJECT)
        -------- (and so on)
        METHODS:
```

4.3.3 Constraint Definition Blackboard Partition

The Constraint Definition BB (CDBB) contains various constraints that are imposed on the designed object. The constraints can be of two types: constraints local to the object (designed) and interaction (interface) constraints that several objects should satisfy simultaneously. An example of a local constraint is *Beam.bending-stress should be less than 0.66*Beam.material.yield-stress*, while the example of an interaction constraint is *Pipes greater than 2 inches cannot go through steel beams or columns*. Only the interaction (or interface) constraints will exist in the CDBB; the local constraints will reside in objects of individual KMs.

An object describing these constraints could be:

```
Constraint-Object
        CONSTRAINS:
                range: (IS-A SBB-OBJECT)
        INTERACTION:
                status:
                range: (IS-A INTERACTION-CONSTRAINT)
        OTHERS: (as needed)
        METHODS:
```

The *status* facet can take values like *satisfied*, *suspended*, *violated*, etc. A taxonomy of these constraints can be defined by the user. Adequate facilities will be provided for the user to incorporate these constraints. For further information on the use of constraints in structural design see [16].

4.4 Knowledge Modules

The Knowledge-base (KB) consists of a number of Knowledge Modules (KMs). Each of these KMs are further decomposed into small units called Knowledge Sources (KSs). The architecture of most KMs could be very similar to the overall architecture of CIDCIS, i.e., knowledge is distributed among several objects (or KSs) and communicate through message passing. KSs can also be decomposed into smaller units, if desired. Thus the KB reflects the hierarchical design process.

Some KMs may incorporate both *textbook* and *heuristic* (surface) knowledge, while other KMs may include *fairly deep* knowledge. Surface knowledge consists mainly of *production rules* encoding empirical associations based on experience. This type of knowledge is useful for setting interface constraints between disciplines and between levels of design interaction. In a system with deep knowledge, both causal knowledge and analytical models would be incorporated. A fully deep system may be difficult to realize with the current state of the art of KBES. However, it is possible to encode analytical models. In this study, the term fairly deep knowledge is used to denote analytical models.

The KMs, although distributed, can be classified into the following categories: *Strategy*, *Specialist*, and *Resource KMs*. These KMs are briefly described below.

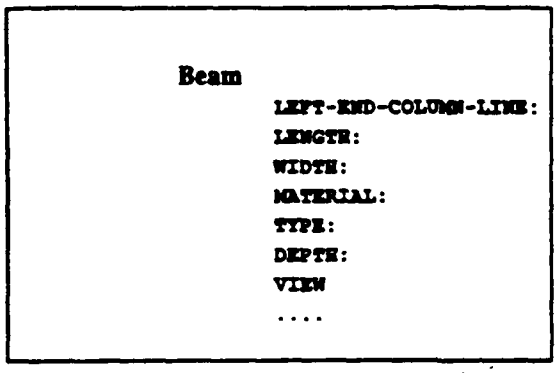
- **Strategy KMs** analyze the current solution state to determine the course of next action. A scenario using the Strategic KM is described in Section 6.2. Since this level may used to control various tasks, such as the activation of Specialist KMs during the coordination process, it comprises the *task control knowledge*.
- **Specialist KMs** contribute to various stages of design and construction. Most KMs at this level are KBES that have a local *Blackboard* which may be divided into various levels of abstraction, and several KSs that interact with the local BB. For example the possible KMs required for an interior finishes design and construction problem are:
  1. *Architectural Designer*, for layout and finishes, including flooring and ceiling systems, etc.
  2. *HVAC*, for heat load calculations, duct layout, diffusers, etc.
  3. *Lighting*, for layout, lighting levels, heat generation, etc.
  4. *Plumbing*, for layout, etc.
  5. *Construction Planning*, for schedules, costs, constructability checking, etc.
  6. *Structural*, only for detailing attachments.

  Individual KMs will, most probably, be residing on different machines and will make extensive use of networking protocols for communicating with the Blackboard.
- **Resource KMs** contain the analytical knowledge and reference information required for analysis and design. These KMs are typically comprised of algorithmic programs and databases. Resource level KMs comprise the algorithmic knowledge of the domain. The Specialist KMs mostly communicate with the Resource KMs through a Blackboard that is local to the Specialist KM.

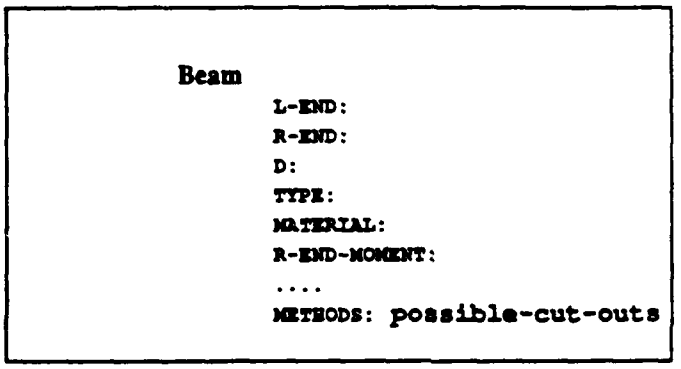
The user forms an integral part of these KMs. An important issue in the development of KMs is the man-machine interface and how the information generated by the user is transmitted to other KSs. We assume that the user interacts with the computer through a high resolution bit mapped display (or appropriate system). Hence, there is a need to provide the appropriate semantic translations from the information provided by user to the form required by other KMs or KSs. In the proposed system, this will be achieved by the interface definition module. Further changes made by the user will be recorded in the local and global Blackboards (if needed) and appropriate actions triggered. Hence, the user can be viewed as a KS taking part in the solution process.

The KMs (mostly Specialist and Strategy) can post and retrieve information from the global Blackboard. However, an object (and associated attributes) in the Blackboard can have varied connotations (most semantic) in

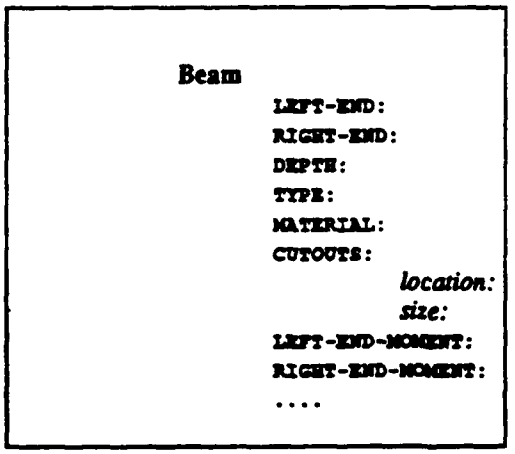
different KMs. Hence, there is a need to define the semantic mappings (translations) between the objects in the KMs to the objects in the Blackboard. As an example, consider the object **Beam**. In a architectural KM, the beam may be defined as follows [8]:



While the same object may be defined in a HVAC KM as:



In the Blackboard, the same object may be defined as:



The methodology used in [8] seems promising adapted for developing the necessary semantic translations in CIDCIS.

5 Current Status

During the first year of the project, our major focus has been the development of: 1) utilities for defining the SBB object hierarchy, 2) transactions for posting, modifying and deleting information from the Blackboard, and 2) a simulation program to demonstrate the utility of CIDCIS, which is being implemented in a hybrid programming environment called FRULEKIT; FRULEKIT supports programming in frames and rules and was developed in LISP at Carnegie-Mellon University by Carbonell and Shell.

These topics are briefly described below.

5.1 Graphic Definition of Objects

The use can interactively define class objects[2] in the Blackboard and KMs. Classes can either be created in LISP or through a menu interface provided to the user. Each class has a name, several slots which describe various attributes, and associated with each slot are facets which provide further information about the slot; the facets also contain methods.

A class object is created by clicking on CREATE-CLASS in a menu on the screen. After creating a class object, the user can display the class using the DISPLAY-CLASS option, as shown in Figure 7.

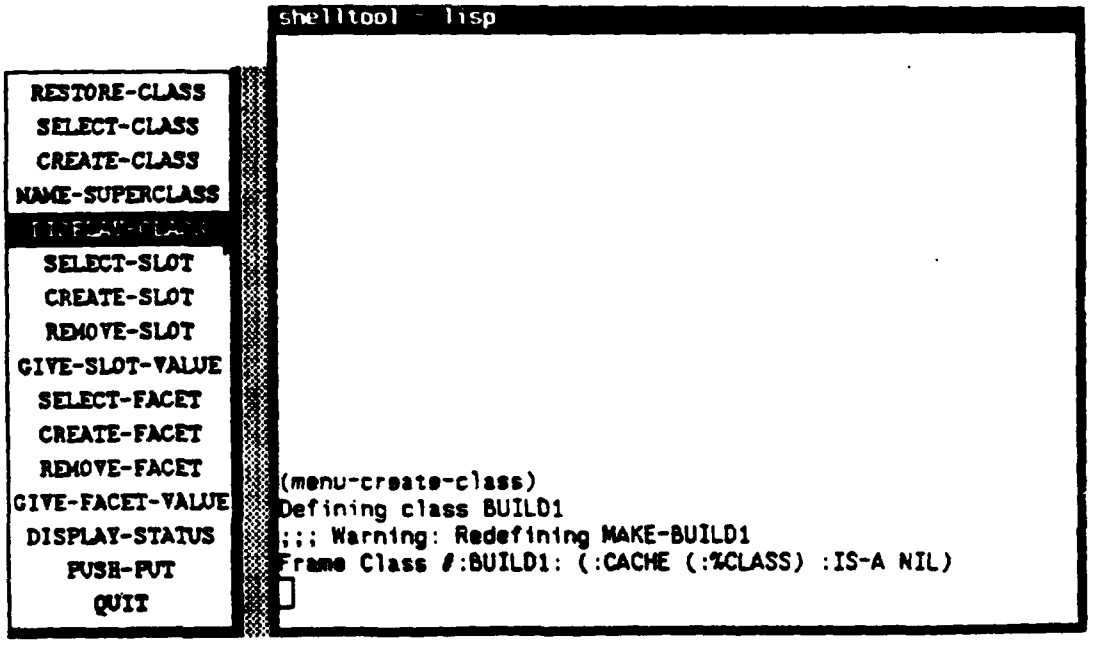


Figure 7: Displaying a Class

In Figure 8, the class **Build1** is made a subclass of the **Hierarchy-object** class, which becomes **Build1**'s superclass. When this link is made, all the slots in the **Hierarchy-object** class are inherited by **Build1**. In addition, the user can create new slots or delete slots. For example, in Figure 9 the user created two slots, namely NAME and HAS-PARTS.

[2] A class denotes the grouping of objects (instance or class) which have similar characteristics, while an instance is a particular individual which belongs to a class.

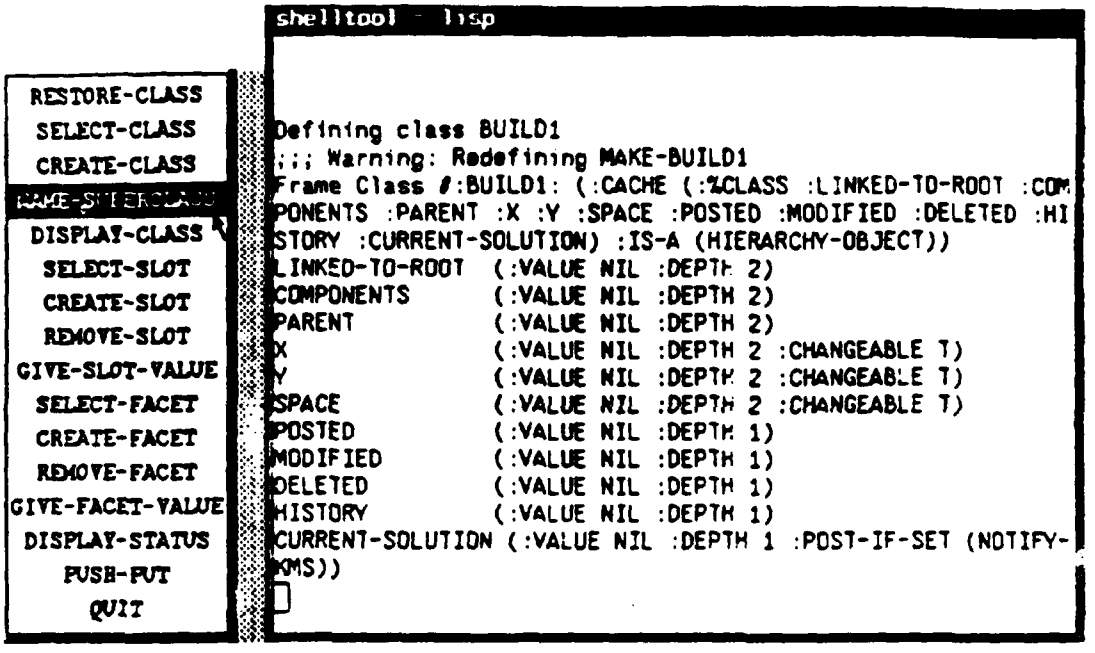


Figure 8: Linking a Class to its Superclass

Slots can be faceted or non-faceted. Facets slots have facets, which provide further information about the slot. The creation of facets is shown in Figure 10.

Instances of a class can either be defined interactively through a menu or can by LISP functions. For example the function *(create-instance 'Build1 'Hyatt-regency)* would create an instance, Hyatt-regency, of Build1.

In addition to defining classes and instances, the user can also display the class hierarchy, in the form a tree. Nodes in the tree depict classes and instances. Each node is displayed in a box with the name of the class or instance. If the name does not fit in the box then it is abbreviated. The user can drag the mouse pointer over the tree to an appropriate node. This will display the full name of the node (Figure 11 a). If the user wants more detail about the node, he can click on the node and he will be shown the slots of the object corresponding to the node (Figure 11 b). Facet information for any slot can be obtained by clicking on the slot (Figure 11 c).

5.2 Blackboard Transactions

Communication betweens KMs is achieved through the Blackboard. The communication channels are established in special slots of the object hierarchy in the SBB. Whenever a new KM is attached to CIDCIS, its address is placed in a special frame in the Coordination Blackboard partition.

Currently, three types of messages can be sent to the Blackboard from KMs (and in some cases vice versa). All messages are put in a mail-box object and processed sequentially. These messages are described below.

1. Post allows a KM to store an object or objects at the appropriate levels in the SBB. The syntax of post is: *(Post local-object remote-object &file)*, where *local-object* is the object or pointer to a tree of objects in a KM, *remote-object* is the level in SBB, *file* is the name of the file that *local-object* is stored in; the & sign indicates that the file name is optional and the system creates its own name if the

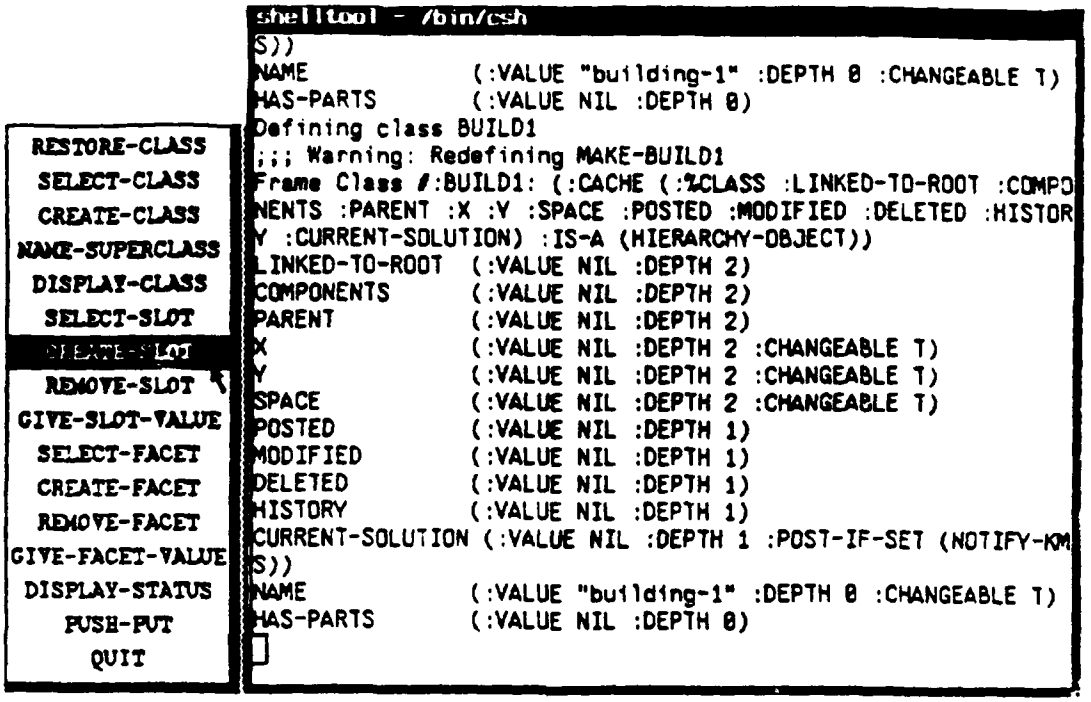


Figure 9: Creation of Slots

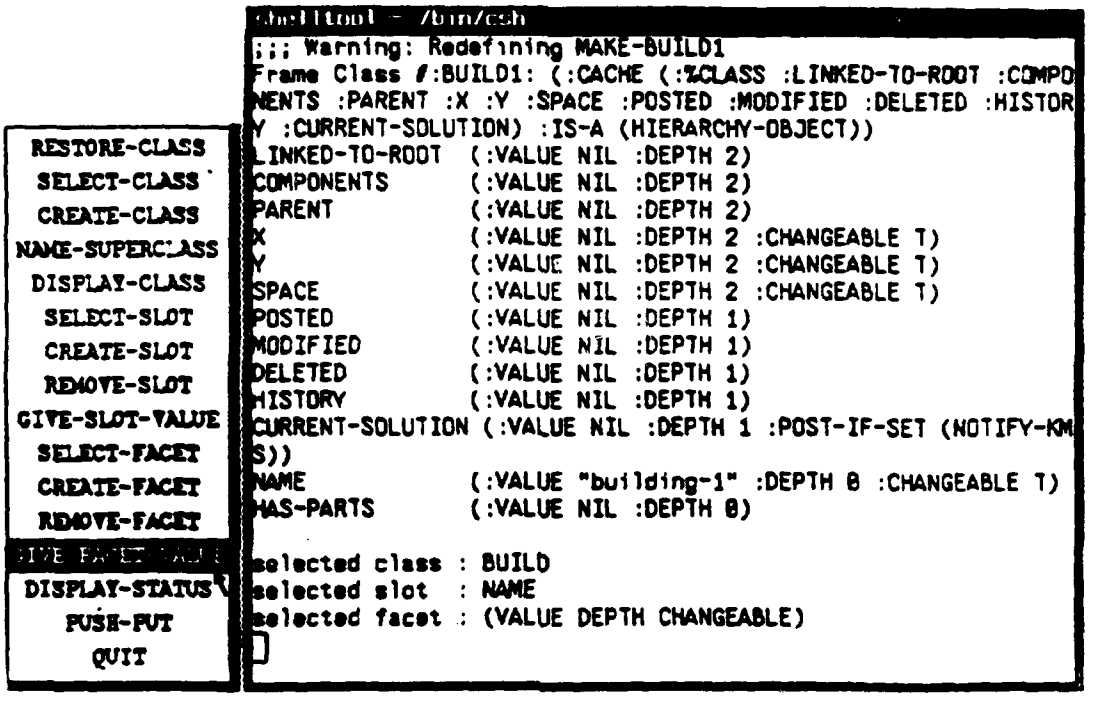


Figure 10: Creation of Facets

file name is not provided. As soon as the Blackboard receives the posted message it accesses the

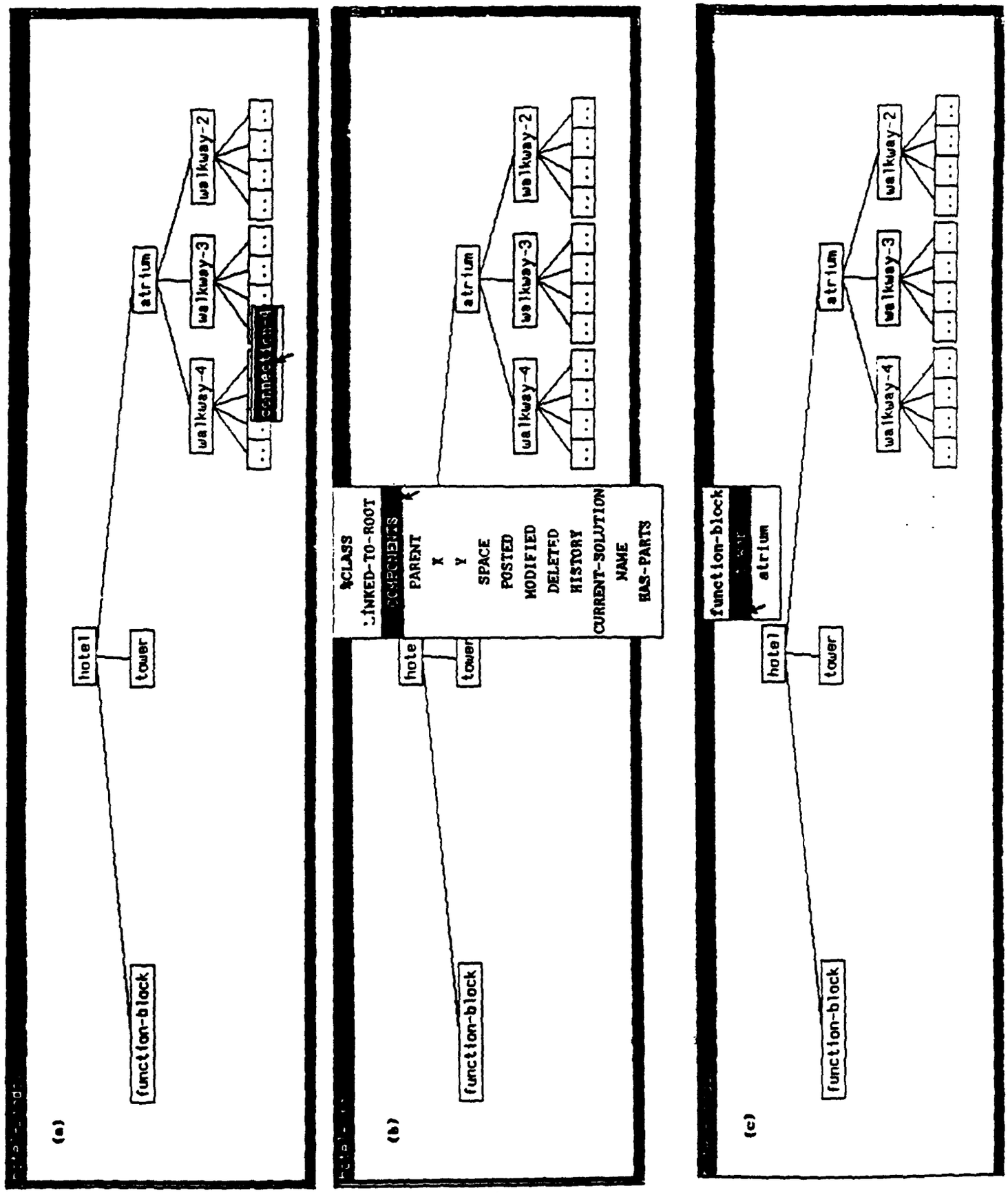


Figure 11: Display of the Object Hierarchy

appropriate *file* in the KM and updates the SBB. This process is depicted in Figure 12.

2. Retrieve gets the information from the SBB to a KM. The syntax of this command is *(Retrieve remote-object &file)*. If object does not contain any information, i.e., it has a value *nil*, in the SBB then the Blackboard relays a message across the network to the appropriate KM that can provide the information; it is assumed that objects in SBB contain the names of the KMs that can update the objects. The retrieving process is depicted in Figure 13.
3. Delete provides a KM the ability to delete information on the Blackboard (SBB). The syntax of delete is *(Delete remote-object)*. Delete does not erase predefined classes, but only removes the instances. This function is currently being updated.

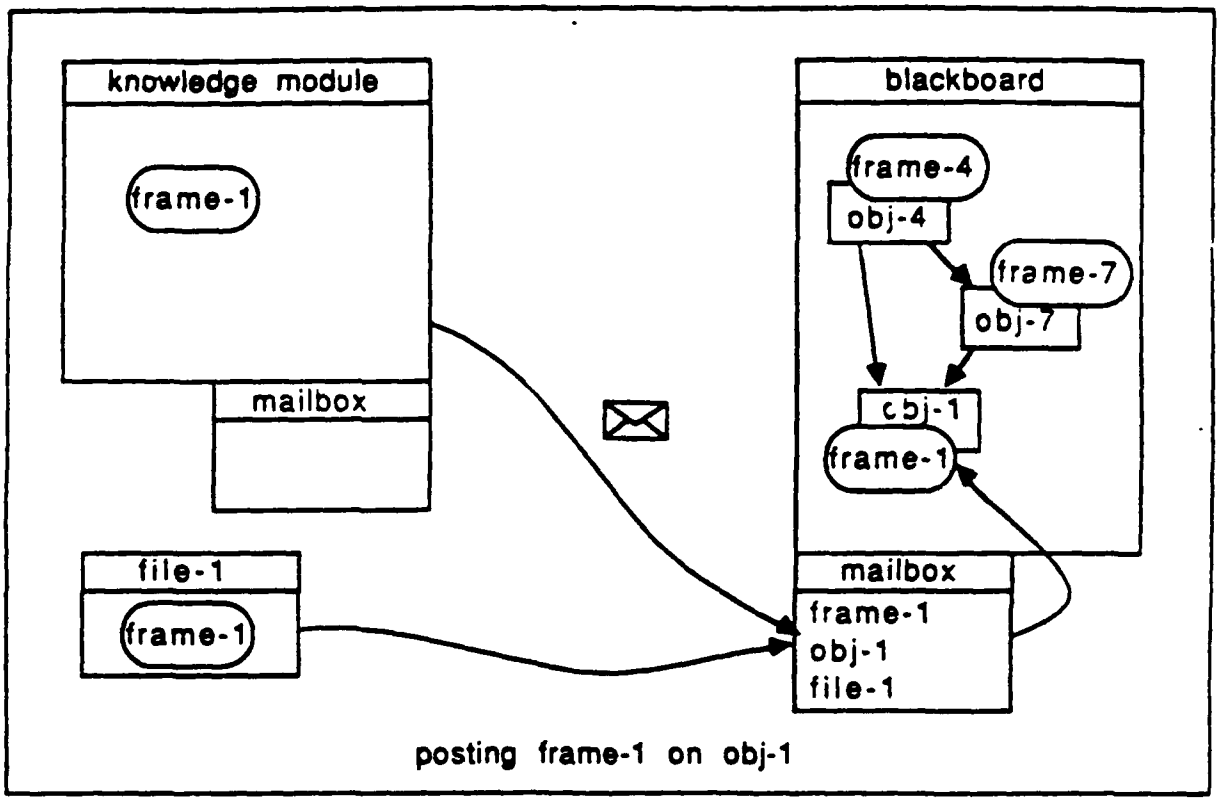


Figure 12: Posting Information to the Blackboard

5.3 A Simulation

A simulation of the Hyatt Regency design process was developed on two SUN computers to demonstrate some of the capabilities of CIDCIS. The Blackboard and the Critic, Constraint Manager, and Strategic KMs exist on the first machine (Figure 14), while the Connection Designer and the Structural Fabricator KMs reside on the second machine (Figure 15).

The design-fabrication sequence is described below and shown in Figures 15 through 20.

1. Connection designer KM posts the connection design (denoted by 1-rod-connection) of the fourth floor walkway on the Blackboard (Figure 15 a).
2. Blackboard receives the design (Figure 16 a and b).
3. The connection object has a method that indicates that the connection design should be checked by the Critic KM. Hence, the Blackboard sends the connection design to the Critic KM (Figure 16 c and d).

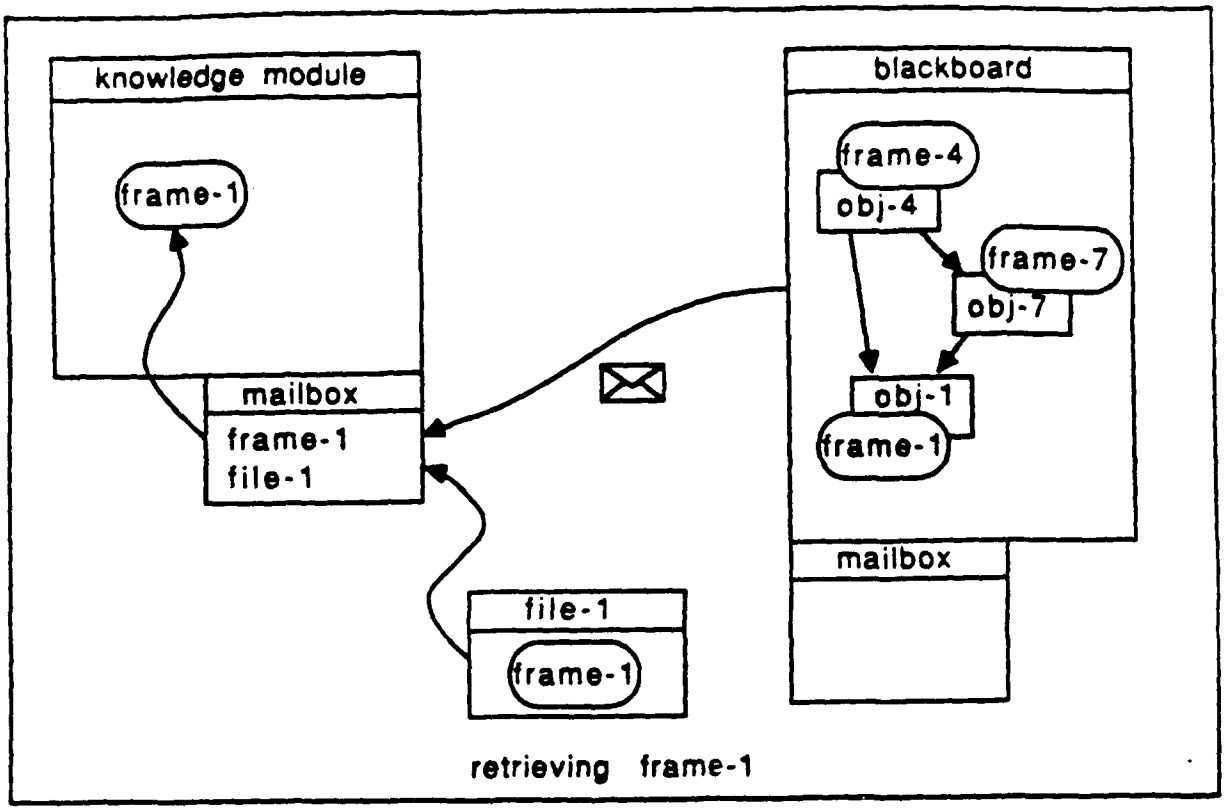


Figure 13: Retrieving Information From the Blackboard

4. The Critic KM replies that the connection design is acceptable[3] (Figure 16 e).
5. The Structural Fabricator KM is sent a message that a connection design has been completed and needs fabrication (Figure 17 a). The Fabricator retrieves the connection design, makes modifications and sends it back to the Blackboard (Figure 18 a, Figure 19 a and b).
6. Blackboard notifies the Strategic KM to check for possible conflicts (Figure 19 c).
7. Strategic KM retrieves the two connection (rod) designs (Figure 19 d) and sends it to the Constraint Manager KM to check for violation of interface constraints (Figure 19 e).
8. Constraint Manager KM notifies the Strategic KM that the designs are incompatible (Figure 19 f).
9. Strategic KM notifies this to both the Connection Designer and the Structural Fabricator (Figure 18 b and c, Figure 20 a and b).

6 List of Publications and Technical Presentations

The current project has some unique features that are not fully integrated in other systems. Some of these features are:

1. Object oriented Blackboard, which acts as an intelligent database.
2. Distributed Computing, where KMs reside in different computers.
3. Heterogeneous KMs, where KMs may be implemented in different programming environments.
4. Negotiation, where conflicts between cooperating KMs are resolved.

We feel that CIDCIS will be instrumental in improving productivity in engineering, reducing design errors,

[3] In the actual design the original connection design itself was faulty, but we assume here that it is an acceptable design.

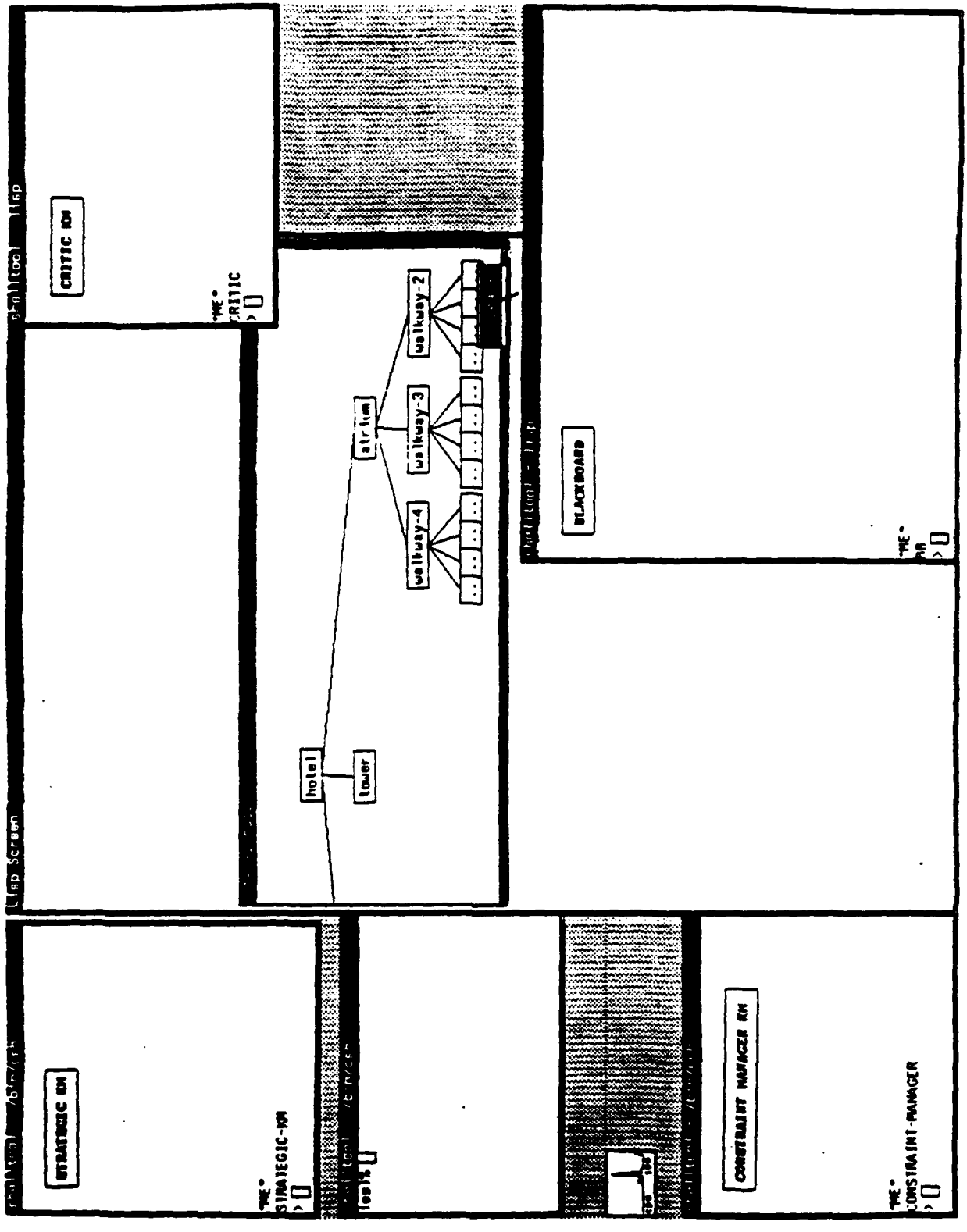


Figure 14: Set up for Simulation of the Hyatt Regency Design Problem

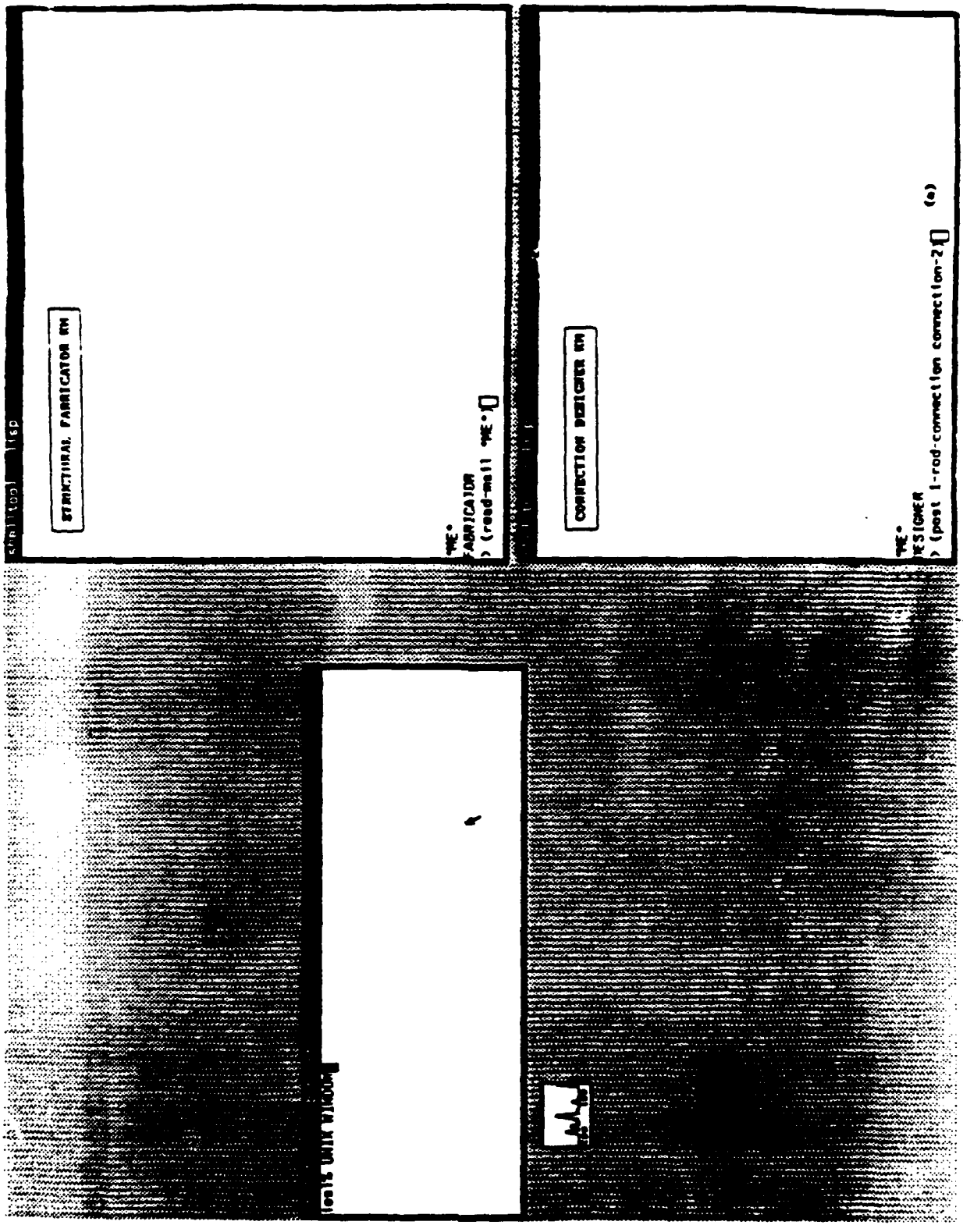


Figure 15: Set up for Simulation of the Hyatt Regency Design Problem (Ctd.)

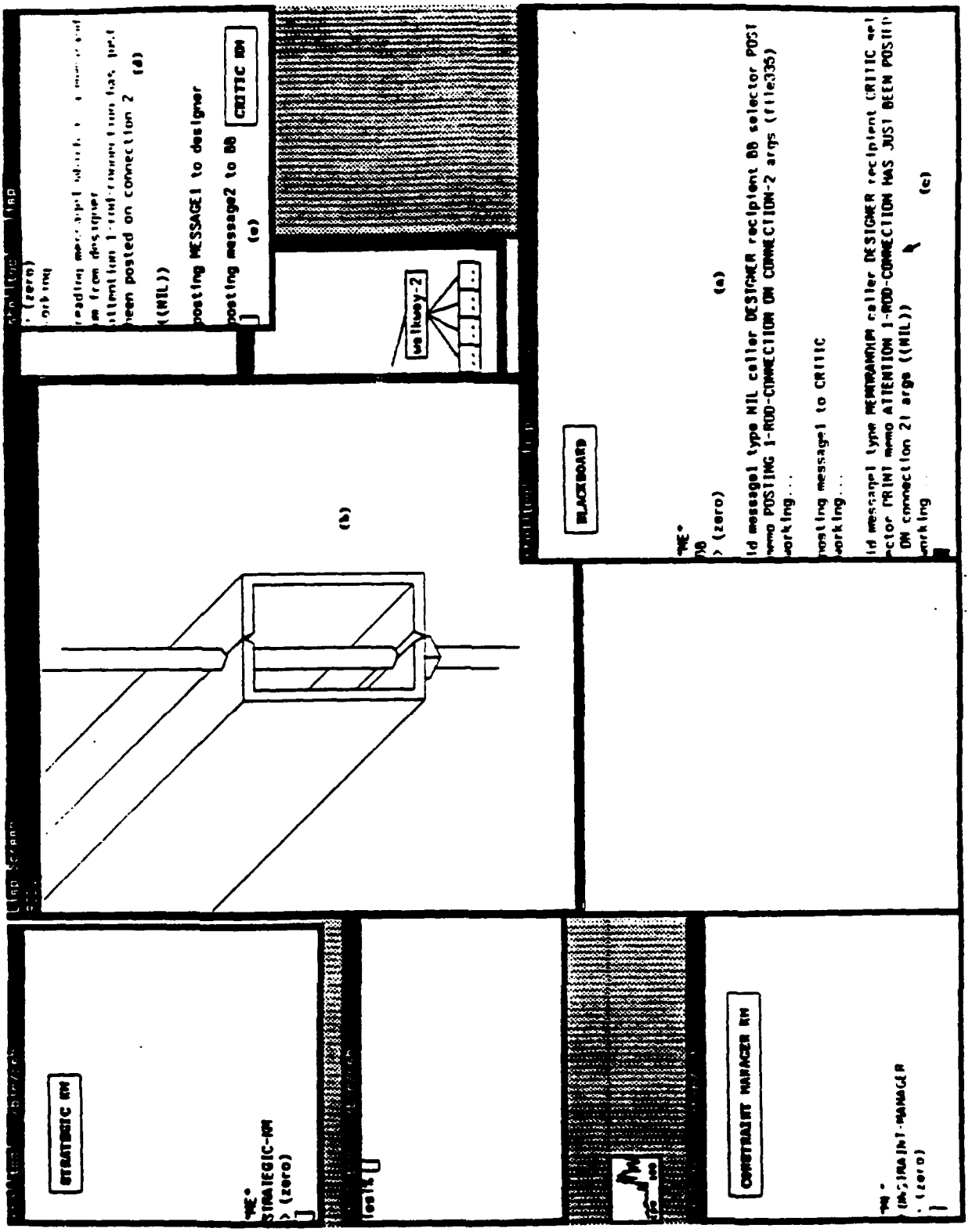


Figure 16: Simulation

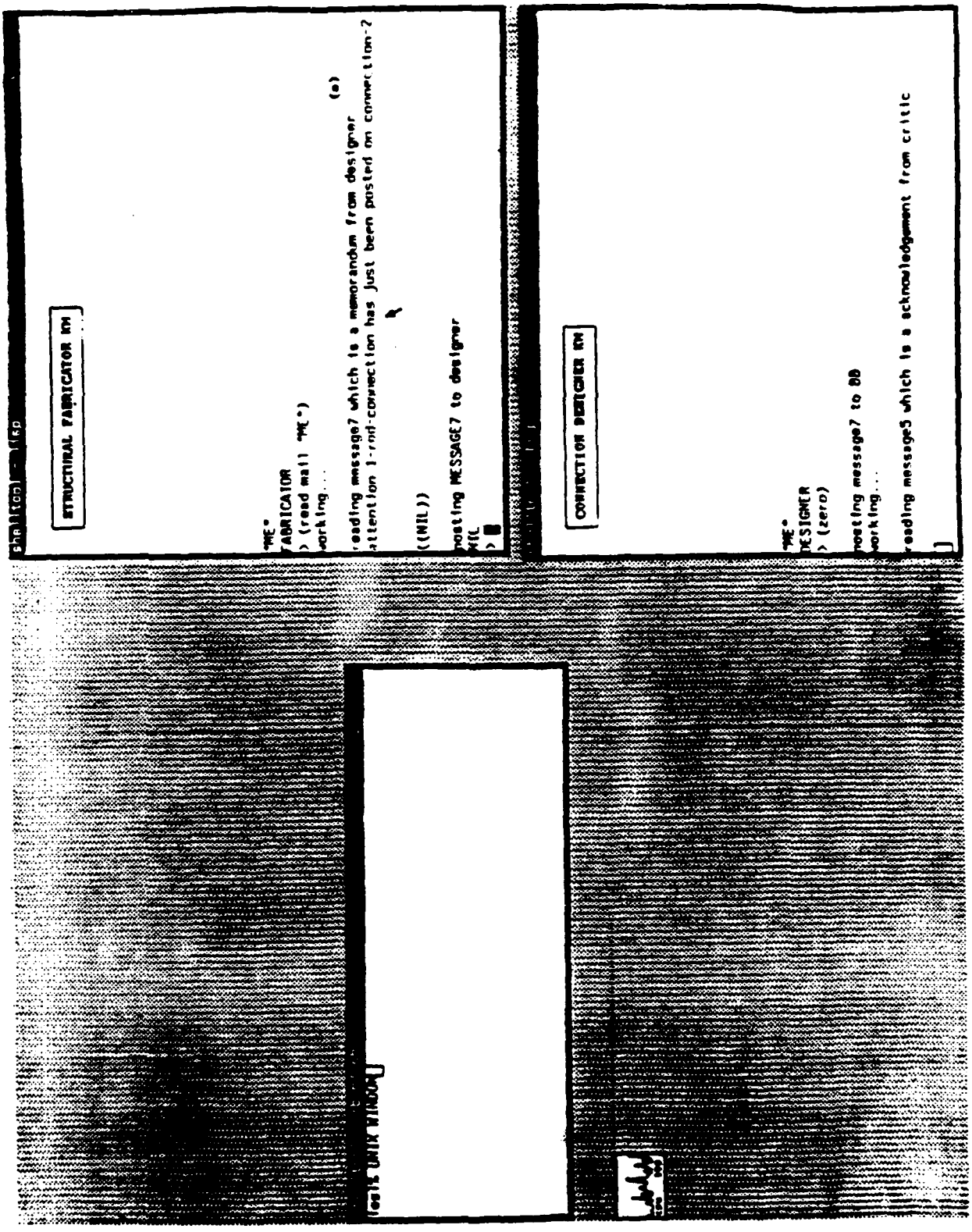


Figure 17: Simulation (Continued)

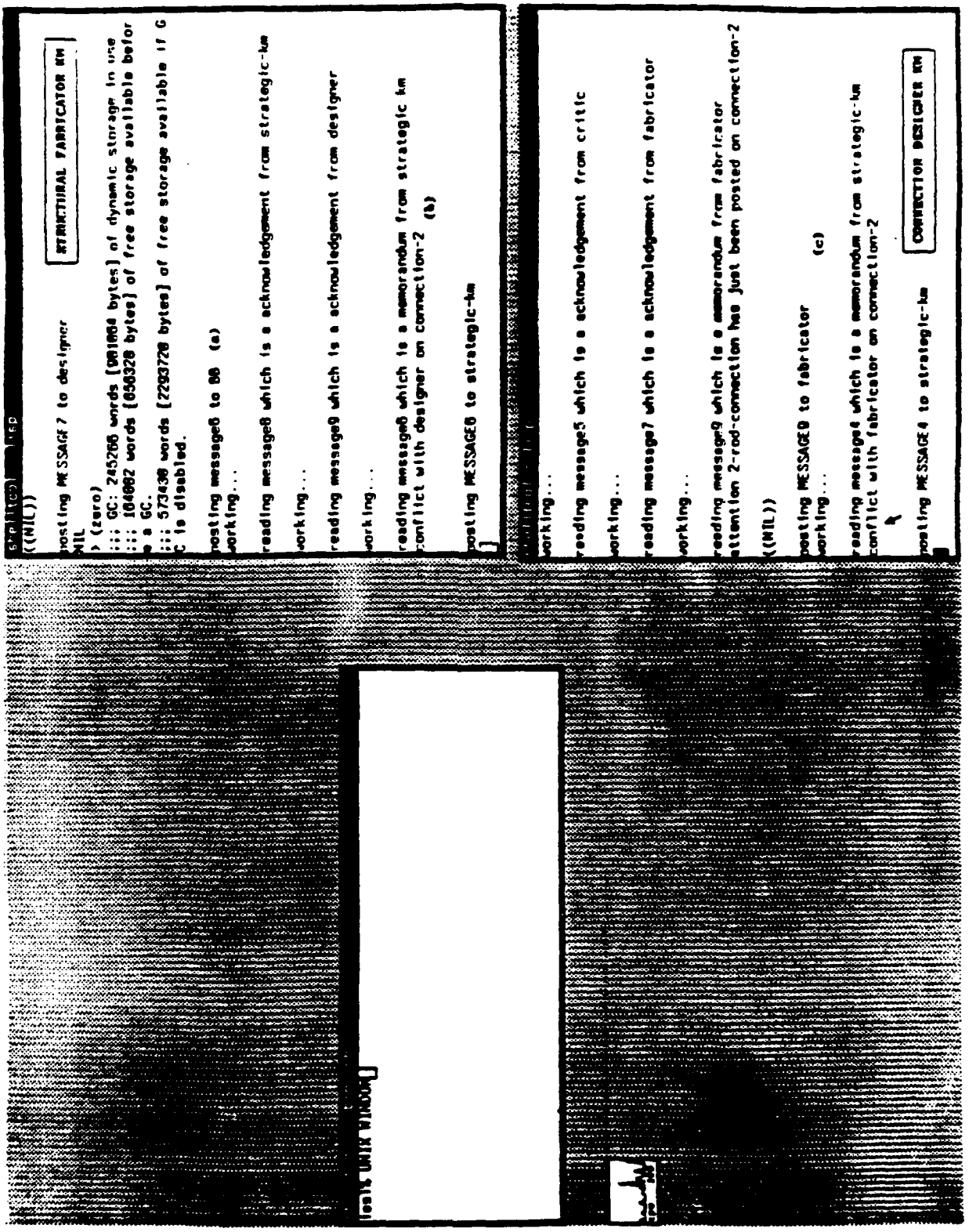


Figure 18: Simulation (Continued)

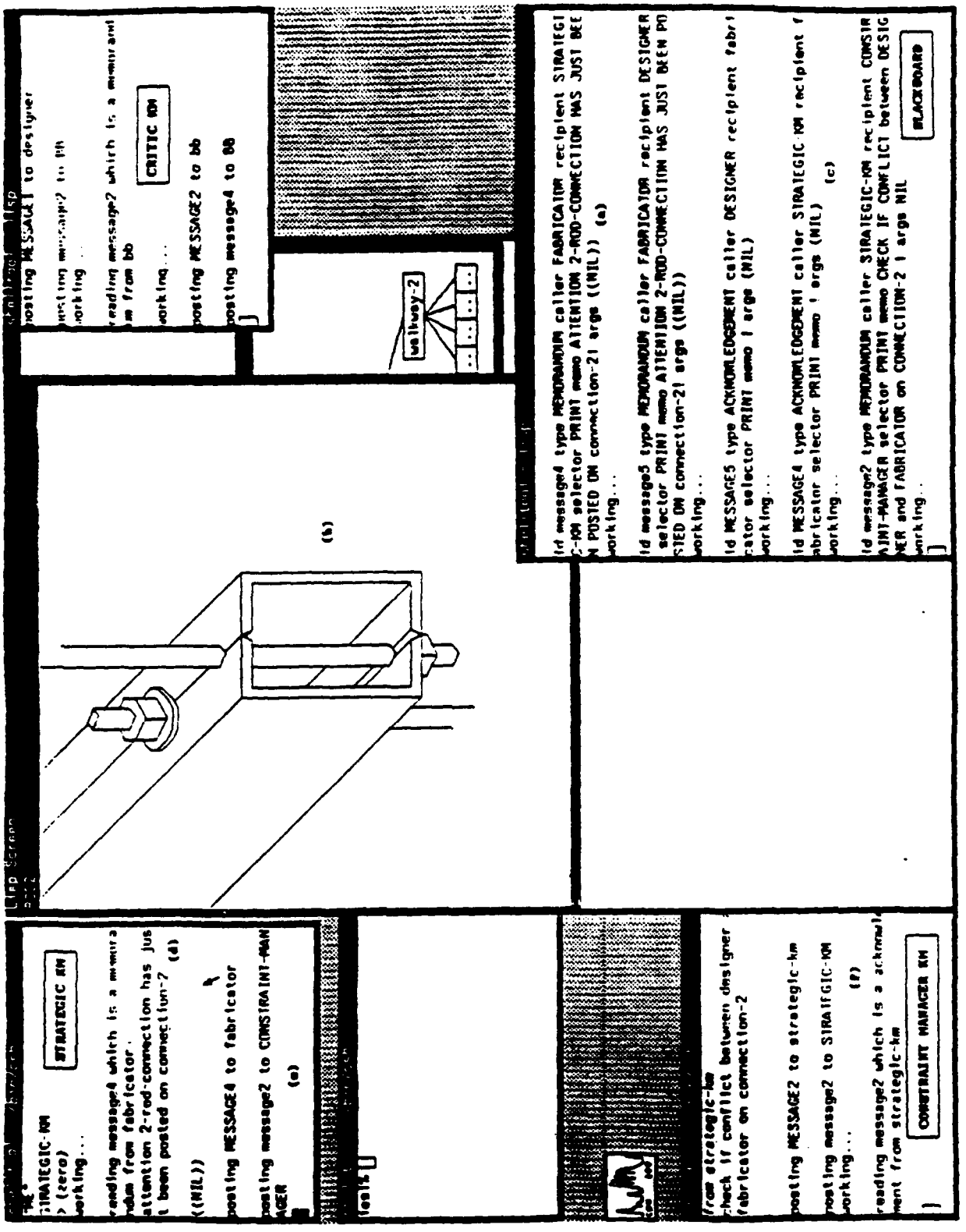


Figure 19: Simulation (Continued)

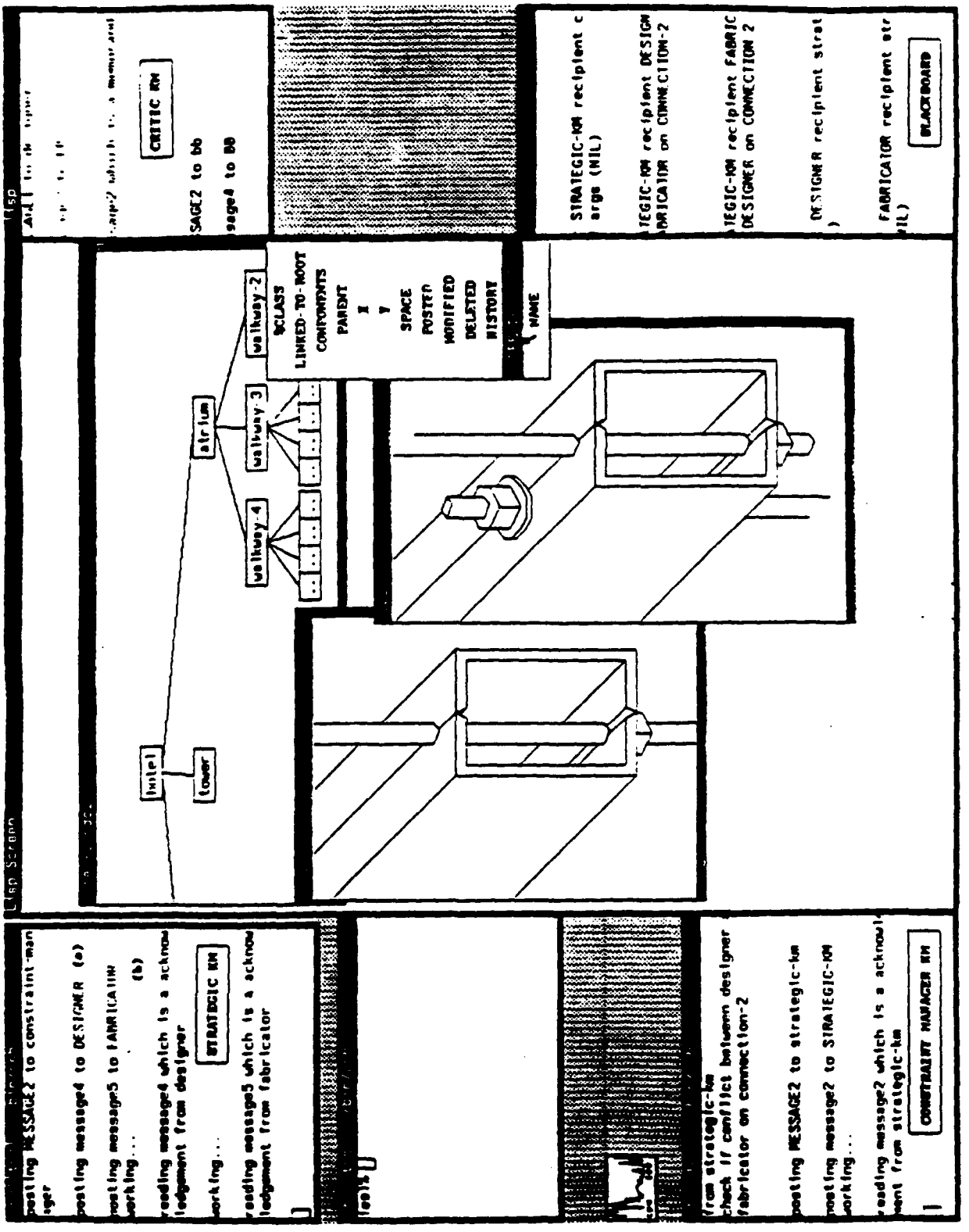


Figure 20: Simulation (Completed)

increasing the detail to which design is carried prior to manufacturing or construction, increasing the speed of the design process, and bringing recognition of the impacts of design decisions on time, cost, and manufacturability or constructability back to the designers. Hence we are making considerable effort to present our work in various conferences. A list of publications based on this work is given below.

1. Sriram, D., Logcher, R. and Groleau, N., *A Framework for a Computer Integrated Design and Construction System* International Conference on Computational Engineering Science, Atlanta, Georgia, April 1988 [Invited].
2. Sriram, D., et al., *Artificial Intelligence in Engineering Design: The M. I. T. Perspective*, Indo-US Systems and Signals Workshop, Bangalore, India, January 1988 [Invited].
3. Sriram, D., *Blackboard Architectures in Engineering: Some Examples*, AAAI Workshop on Blackboard Systems, American Association of Artificial Intelligence, Philadelphia, July 1987.

In addition, the support of ARO is also acknowledged in the following books:

1. Tong, C. and Sriram, D. (Editors) *Artificial Intelligence Approaches for Engineering Design*, 1988 (Forthcoming).
2. Sriram, D. (Editor) *Knowledge-Based Expert Systems For Engineering*, 1988 (Forthcoming).

7 Summary

The development and testing of knowledge based computer tools for the integration of design and construction (CIDCIS) were described. CIDCIS is intended to provide coordination among multiple designers impacts of design decisions. It can be envisioned as a network of computers and users (called Knowledge Modules), where the communication and coordination is achieved, through a global database - the Blackboard, by a control mechanism. During the first year of the project our major focus has been the development of: 1) utilities for defining the object hierarchy in the Blackboard, 2) transactions for posting, modifying and deleting information from the Blackboard, and 2) a simulation program to demonstrate the utility of CIDCIS.

8 Bibliography

[1] Amarel, S., "Basic Themes and Problems in Current AI Research," *Proceedings of the Fourth Annual AIM Workshop*, Ceilsielske, V. B., Ed., Rutgers University, pp. 28-46, June 1978.

[2] Barton, P. K., *Building Services Integration*, E. F. N. Spon, 733 Third Ave., NY 10017, 1983.

[3] de Kleer, J., "Choices Without Backtracking," *Proceedings of the 4th NCAI*, Texas, AAAI, August 1984.

[4] Doyle, J., "A Truth Maintenance System," *Artificial Intelligence*, Vol. 12, pp. 231-272, 1979.

[5] Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R., "The Hearsay-II Speach Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys*, Vol. 12, No. 2, pp. 213-253, 1980.

[6] Hayes-Roth, B., "A Blackboard Architecture for Control," *Artificial Intelligence*, Vol. 26, No. 3, pp. 251-321, 1985, [Also Technical Report No. HPP 83-38, Stanford University.].

[7] Hayes-Roth, B., *The Blackboard Architecture: A General Framework for Problem Solving*, HPP 83-30, Dept. Computer Science, Stanford University, May 1983.

[8] Howard, H.C., *Integrating Knowledge-Based Systems with Database Management Systems for Structural Engineering Applications*, unpublished Ph.D. Dissertation, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, 1986, [See also Special Issue of CAD, November 1985.].

[9] Jagannathan, V., et al., "Workshop on Blackboard Systems: Implementation Issues," 1987.

[10] Katz, R., *Information Management for Engineering Design*, Springer-Verlag , 1985.

[11] Marshall, R. D., et al., *Investigation of the Kansas City Hyatt Regency Walkways Collapse*, Technical Report Science Series 143, National Bureau of Standards, Washington, D. C., May 1982.

[12] Nii, P. and Brown, H., "Blackboard Architectures," AAAI Tutorial Notes No. HA2, 1987.

[13] Nii, P., Fiegenbaum, E., Anton, J. and Rockmore, A., "Signal-to-Symbol Transformation: HASP/SIAP Case Study," *The AI Magazine*, Vol. 3, No. 2, pp. 23-35, Spring 1982.

[14] Rehak, D.R., Howard. H.C., and Sriram, D., "Architecture of an Integrated Knowledge Based Environment for Structural Engineering Applications," in *Knowledge Engineering in Computer-Aided Design*, Gero, J., Ed., North Holland, 1985.

[15] Rehak, D., "CAD in Design and Construction," Dept. of Civil Engineering, Carnegie-Mellon University, 1986.

[16] Sriram, D. and Maher, M. L., "Representation and Use of Constraints in Structural Design," *AI in Engineering*, Adey, R. and Sriram, D., Eds., Southampton, UK, CML Publications, April 1986, in press.

[17] Sriram, D., *Knowledge-Based Approaches for Structural Design*, CM Publications, UK, 1987.

[18] Sriram, D., Ed., *Knowledge-Based Expert Systems for Engineering*, Forthcoming, 1988.

[19] Stefik, M. and Bobrow, D., "Object-Oriented Programming: Themes and Variations," *The AI Magazine*, Vol. Winter, pp. 40-62, 1985.